2008 年度
芝浦工業大学　システム工学部
電子情報システム学科

総合研究論文

衛星観測データのデジタルプラネタリウム用コンテンツにおける画像処理
～「あかり」衛星の見た宇宙～

Image processing in contents for digital planetarium on the astronomical data

～The universe from the satellite 'AKARI'～

P05030

小倉（おぐら） 博之（ひろゆき）

指導教員：久保田 あや

目 次

第1章　はじめに

宇宙の様子は可視光で見えるだけではなくX線や赤外線といった多波長の電磁波の観測を行うことで、天体の現象、宇宙の起源などを知ることができる。日本で開発された天文衛星「すざく」や「あかり」は現在もそのような観測を続けている。天文衛星から入手した画像データは研究者の中では広く利用されているものの、一般の人に活用される機会は少ない。

現在は画像や動画などのデジタルデータを全て計算機で管理、運営できる「デジタルプラネタリウム」という、新しい形のプラネタリウムが普及してきている。デジタルデータならば可視光以外の多波長の観測の様子も簡単に投影することができ、プラネタリウムという媒体を通して一般の人に理解してもらう機会が増えるだろう。しかしその投影には決まったフォーマットがあるため、専用の形式へ変換する作業が必要となる。よって本総合研究として、赤外線天文衛星「あかり」のデータを基に、宇宙研と共同で行っている天体観測データをデジタルプラネタリウム投影用のフォーマットへ変換する。またこの過程で画像処理を導入することで画像としての質の向上をはかる。

第2章　デジタルプラネタリウムとfits画像

そもそもデジタルプラネタリウムとは従来のプラネタリウムとどのような相違点があるのか文献 [1] にしたがって比較する。

「プラネタリウム」は、18 世紀にオランダのアイス・アイジンガが、太陽系の惑星が太陽を中心として楕円軌道ををを周回している様子を再現する装置を製作し、「Planetarium」と呼んだことが語源となっている [1]。今日では、全天周の星空をドーム型スクリーンに再現する装置の星空投映機、あるいは投映機と共にプラネタリウム番組の上映に用いられる音響光学装置を含む設備一式や、その上映設備を備えた建物自体のことを表す言葉として「プラネタリウム」は用いられている。

2.1　光学機械式プラネタリウムとフルデジタルプラネタリウム

2.1.1　光学機械式プラネタリウム

光学機械式プラネタリウムはダンベルのような形をした星空投影機で、両端の球状の恒星投影機に光学レンズが多数備えられている。その恒星投影機結ぶ取っ手の部分に太陽、月、太陽系の惑星を投影する惑星投影機で構成されている。また、串団子のような形のものもあり、それは恒星投影機と惑星投影機の配置が逆になっている。

光学機械式では、恒星投影機を機械的に回転または移動させて、北極星を中心とした恒星の日周運動をと年週運動を再現している。恒星投影機は 32 面体の形状をしており、電球と恒星原板、レンズから成り立っている。恒星原板とは、恒星の位置と明るさを正確に反映した小さな穴の開いている板を指す。恒星投影機の中心部にある電球からの光は、32 面体の各面に配置された恒星原板とレンズを通過した後、スクリーンに恒星の光として映し出される。また、公転などにより生じる太陽系惑星の夜空での動きを、複雑に組み合わせた歯車の動きにより細かく姿勢を制御することにより再現している。光学機械式プラネタリウムとは、天体の位置計算と表示に特化した一種のアナログ計算機であると言えるだろう。

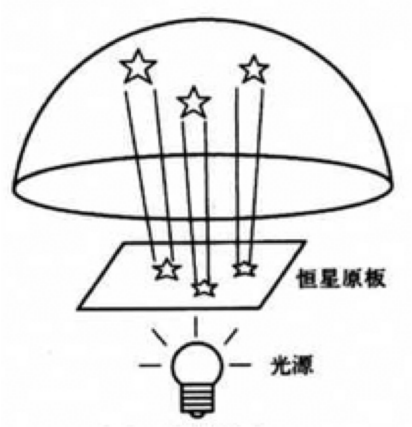

図 2.1: 光学機械式プラネタリウム　出典：文献 [1] の図 1 (a) より

2.1.2 フルデジタルプラネタリウム

フルデジタルプラネタリウムとは計算機で星空あるいは宇宙空間の画像を生成し、大画面用プロジェクタに出力し、魚眼レンズを介してドーム型スクリーンに映し出す形式の星空投映機のことを指します。これにより、光学機械式で行っていた歯車の組み合わせを用いた天体の位置計算を、ソフトウェアでリアルタイムに求めることが可能となった。フルデジタル式での、星空あるいは宇宙空間の画像の作成と投映の手法は 2 種類あり、

- 解説者の演出に応じて、三次元の CG で星空をリアルタイムに作成して投映する「生番組」方式
- 予め作成した星空の画像を、上映前に動画像として圧縮と符号化を行い、上映時に再生して投映する「録画番組」方式

である。光学機械式では、星空、静止画、動画、音声といった上映番組を構成するコンテンツごとに再生のための専用ハードウェア（星空投映機、スライド、ビデオ、カセットデッキなど）を用意し、各ハードウェアの再生を制御する必要があった。それに比べてフルデジタル方式では、全てのコンテンツをデジタルデータ形式に変換して計算機のハードディスク上に格納し、それらの再生をソフトウェアでプログラム化している。これにより、上映手順の自動化が可能となった。

図 2.2: フルデジタル式プラネタリウム　出典：文献 [1] の図 1 (b) より

2.2 天空座標の表し方

地球上には世界地図というものがあるように、宇宙にも天体や銀河などがどこにあるかを知るために地図状の表現がある。地球では緯度、経度が用いられるが、宇宙空間ではどこを基点とするかによって様々な座標系が存在する。ここでは我々の視点から見た赤道座標と銀河を中心として見た銀河座標について紹介する。

2.2.1 赤道座標と銀河座標

赤道座標

赤道座標は、星の絶対位置をあらわすために用いられる、地球の自転を基準とした座標系で、赤経、赤緯と呼ばれる 2 つの数値であらわされる。地球の自転軸を北に伸ばし天球と交わる点を「天の北極」、南に伸ばし天球と交わる点を「天の南極」とし、これに地球上の経度、緯度線をそのまま天球上に貼りつけたものと考えるとよい。一般にプラネタリウムに投影されている宇宙空間の様子はこの座標系で表現されている。赤緯は赤道面を基点（0 °）とし、南（-）北（+）にそれぞれ 90 °までの数値で表す。赤経は春分点（太陽が天の赤道を南側から北側へ横切る点）を基点に、東回りに計り、15 °=1 時、15'=1 分、15"=1 秒として 24 時までの数値で表す。

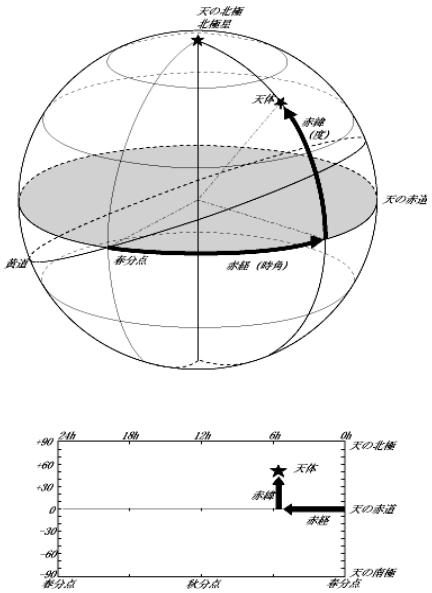

図 2.3: 赤道座標

銀河座標

銀河座標は銀河系内の天体の分布や運動をあらわすときに用いられ、銀河面と呼ばれる基準面（ほぼ天の川の流れの中心に沿って全天を 1 周している）を基点とした座標系である。銀河座標の経度は銀経、緯度は銀緯と呼ばれています。銀経はいて座にある強力な電波源いて座 A と、銀河北極とを結ぶ大円と銀河面との交点を基点（0 °）として東回りに 360 °までの数値で表される。銀緯は銀河面を基点（0 °）とし、南（-）北（+）にそれぞれ 90 °までの数値で表す。

2.2.2 赤道座標から銀河座標への変換

文献 [2] により、赤道座標と銀河座標の変換方法[1]を簡単に説明する。銀河中心の赤経赤緯は（266.40500D,-28.9361D）である。z 軸の周りの ϕ=266 °.40500、y 軸の周りの θ=28 °.93617 の回転で、新たな x 軸が銀河中心を指すことがわかる。しかし、それだけでは銀河面の傾きが決まっていない。さらに「新たな x 軸の周り」で ψ=58 °.59866 回転してやれば、正しく銀河座標が定義

[1]http://plain.isas.jaxa.jp/ ebisawa/TEACHING/2007Komaba/2007Komaba/node16.html

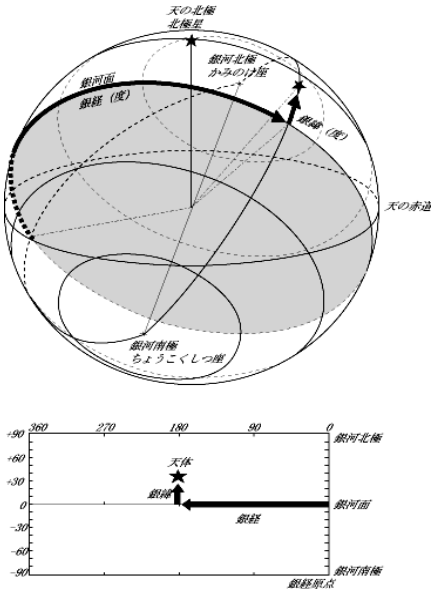

図 2.4: 銀河座標

されることが分かっている。銀河座標系の基底ベクトルをダッシュ付きで表すと式より

$$\begin{pmatrix} e'_x & e'_y & e'_z \end{pmatrix} = \begin{pmatrix} e_x & e_y & e_z \end{pmatrix} \begin{pmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.1}$$

逆変換は以下のようになる。

$$\begin{pmatrix} e_x & e_y & e_z \end{pmatrix} = \begin{pmatrix} e'_x & e'_y & e'_z \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & \sin\psi \\ 0 & \sin\psi & \cos\psi \end{pmatrix} \begin{pmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.2}$$

ここで ϕ, θ, ψ にそれぞれ値を代入し、計算すると、

$$\begin{pmatrix} e_x & e_y & e_z \end{pmatrix} = \begin{pmatrix} e'_x & e'_y & e'_z \end{pmatrix} \begin{pmatrix} -0.0548755 & -0.873437 & -0.483835 \\ 0.49411 & -0.44483 & 0.746982 \\ -0.867666 & -0.198076 & 0.455984 \end{pmatrix} \tag{2.3}$$

となる。

よって、赤道座標での方向ベクトルの 3 成分を（x, y, z）、銀河座標での成分を（x', y', z'）としたとき、

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} -0.0548755 & -0.873437 & -0.483835 \\ 0.49411 & -0.44483 & 0.746982 \\ -0.867666 & -0.198076 & 0.455984 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{2.4}$$

また赤道座標から銀河座標への変換を与える。また、銀河座標から赤道座標への変換を得るには、式の転置行列をとれば良いので、

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} -0.0548755 & -0.494110 & -0.867666 \\ 0.8734370 & -0.44483 & -0.198076 \\ -0.4838350 & -0.746982 & 0.455984 \end{pmatrix} \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} \tag{2.5}$$

となる。

2.2.3 投影法

§2.2.1 では座標系についての記述であるが、デジタルプラネタリウムにおいて全天画像を 2 次元地図に投影するには様々な投影法がある。ここでは典型的な投影法として、AIT、CSC、CAR、TSC について説明する。

AIT：Hammer-Aitoff 投影法

ランベルト正積方位図法の横軸図を変形して擬円筒図法のような形状にしたものである。角の歪みが非常に大きくなっている。Aitoff が ARC(Zenithal equidistan 投影法) を元に考えだしたものに Hammer が修正を加えたものである。この投影法の定義式を以下に示す。

$$x = 2\gamma \cos\theta \sin\frac{\phi}{2} \tag{2.6}$$

$$y = \gamma \sin\theta \tag{2.7}$$

$$\gamma = \frac{180\ ^\circ}{\pi}\sqrt{\frac{2}{1+\cos\theta\cos(\frac{\phi}{2})}} \tag{2.8}$$

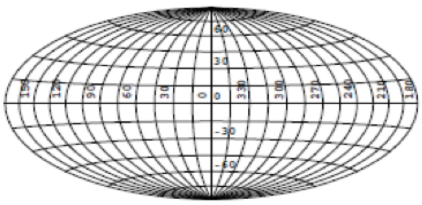

図 2.5: Hammer-Aitoff 投影法

CSC：COBE(COsmic Background Explorer) Quadrilateralized Spherical Cube 投影法

この投影法は図のように極方向の図とそれ以外（赤道方向）とに分けられており、この展開図を組み立てると立方体の内側に全天画像を貼付けたようになる。定義式を以下に示す。

$$x = \phi_c + 45\ ^\circ F(X, \psi) \tag{2.9}$$

$$x = \theta_c + 45\ ^\circ F(\psi, X) \tag{2.10}$$

$$F(X,\psi) = \chi\gamma^* + \chi^3(1-\gamma^*+\chi\gamma^2(1-X^2))[\Gamma + (M-\Gamma)\chi^2 + (1-\psi^2)\sum_{i=0}^{\infty}\sum_{j=0}^{\infty} C_{ij}\chi^{2i}\psi^{2j}]$$
$$+ \chi^3(1-X^2)[\omega_1 - (1-X^2)\sum_{i=0}^{\infty} D_i X^{2i}] \tag{2.11}$$

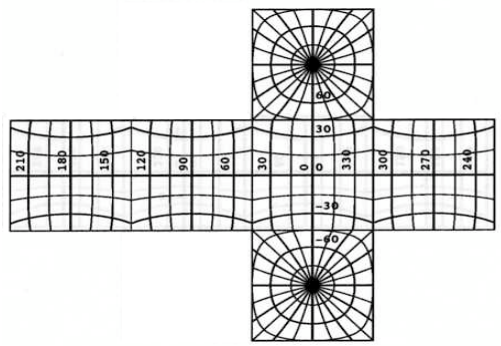

図 2.6: COBE quadrilaterralized spherical cube

CAR：Plate Carree 投影法

経緯線が直角、等間隔である正距円筒図法の 1 つで、標準緯度を 0 度においたものである。この図法は経度、緯度をそれぞれ図の縦と横にそのまま読み替えたもので、標準緯線上と縦方向に関して正距である。標準緯線から離れる程、横方向の長さが拡大されるため、角度は正しくない。定義式は以下である。

$$x = \phi(\text{経度}) \tag{2.12}$$

$$y = \theta(\text{緯度}) \tag{2.13}$$

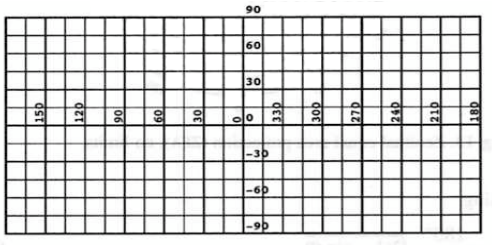

図 2.7: Plate Carree 投影法

TSC：Tangential Spherical Cube 投影法

これは CSC と投影の形は同じで組み立てると立方体の形になる。CSC と比べると天球の割当に対して歪みが少ない。

$$x = (x - \phi_c)/45\ ^\circ \tag{2.14}$$

$$x = (y - \theta_c)/45\ ^\circ \tag{2.15}$$

$$\zeta = 1/sqrt1 + \chi^2 + \psi^2 \tag{2.16}$$

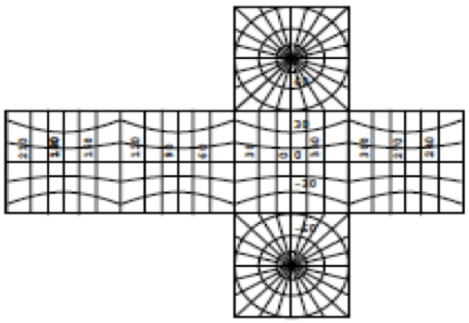

図 2.8: Tangential Spherical Cube 投影法

2.3 FITS ファイルと World Coodinate System

デジタルプラネタリウム投影については、使用されるデータ保存の規格に決められたものがある。また、観測データの情報を読み込み天球座標とデータ配列を対応づけるサブルーチンがあり、これらはデジタルプラネタリウム投影用変換には重要なものとなる。ここではその保存用ファイル形式とサブルーチンについて記述する。

2.3.1 FITS：The Flexible Image Transport System について

FITS(The Flexible Image Transport System)[3] は天文分野の研究者をはじめ、天文アマチュアの間でも画像やデータ、テーブルなどの保存に使われる代表的なデータフォーマットである。
天文現象は長時間に行われるものも多く、そのような現象の解明のためには観測データの交換やアーカイヴ作成が問題なくできるように、きちんとした規格でデータを保存する必要がある。こうしたデータの規格には「互換性」、「単純さ」、「拡張性」、「自己記述性」といった特質が求められる。それに応える形式が FITS である。
FITS は単なる画像フォーマットではなく、天文分野において科学的データセットの運搬、解析、アーカイヴ等、あらゆる場面で FITS ファイルが使われている。例えば、

- 多次元データ配列：1 次元スペクトル、2 次元イメージ、3 次元以上のデータキューブ等で利用
- 様々な情報うを行・列に並べた表形式のデータに利用
- データに関する詳細な情報をヘッダに書いてデータと一緒に供給

といった具合である。

2.3.2 FITS の構造

単純な FITS ファイルの構造は ASCII テキストで書かれたヘッダとバイナリ（通常は多次元の）データ配列からできている。現在ではこの「基本」FITS 要素に加えて同じでーた格納構造（ヘッ

ダ＋データ）を持つ拡張された他の FITS 要素が付け加わっても良いことになっている。模式図で表すと次のようになる。

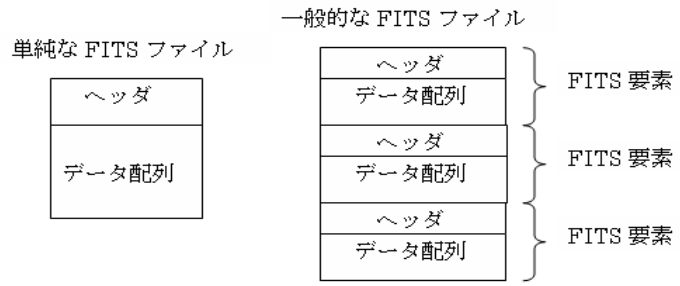

図 2.9: fits 模式図 出典：文献 [3] の p7 図より

多くの FITS ファイルは単純な 1 つのヘッダと 1 つのデータ配列を持つファイルだが、もっと複雑なデータを詰め込むこともでき、実際、最新の大型観測装置が吐き出すデータにはそのようなものも見られるようになっている。FITS ファイルの構成をもう少し詳しく見ていくと、

・（基本）FITS 要素
・（拡張）FITS 要素 1
・（拡張）FITS 要素 2
…

のように（ヘッダ+データ）の FITS 要素が連なっており、いずれの FITS 要素も整数個の（論理）レコードからなる。論理レコードのサイズは 2880 バイト (23040 ビット＝あらゆる計算機のワード長の最小公倍数) である。

1 つの FITS 要素は大きく 2 つの部分に分けられる。

前半部のヘッダはヘッダレコードとも呼ばれ、データの目的、種類、構造、バイト数、レコード数などのデータに関する解説部分となっている。1 行が 80 文字にからなるカードイメージの連なりで、整数この論理レコードに収められる 1 つの行の各欄の使い方や用語には一定の規約がある。

後半部のデータ（データ配列）はデータレコードとも呼ばれ、ヘッダレコードの直後のレコードから実際のデータが書き込まれる。すなわち、1 つの FITS 要素の構造は表のようになっている。

・ヘッダレコード（2880 バイト × n（整数））	ヘッダ 1（80 バイト・カードイメージ） ヘッダ 2（80 バイト・カードイメージ） ヘッダ 3（80 バイト・カードイメージ） …
・データレコード（2880 バイト × n（整数））	データ（バイナリ or アスキー） …

表 2.1: FITS 要素の構造　出典：文献 [3] の p8 図より

2.3.3 基本 fits ファイルでの変換

ROSAT や「あかり」の FITS ファイルは、データ配列のインデックス（i, j, k, ...）から物理量である座標値（x_i, x_j, x_k, ...）への変換のてめに以下のキーワードが定義されている。

CRVALn	参照点での座標値
CRPIXn	参照点でのインデックス
CDELTn	参照点での座標値の増分
CTYPEn	座標軸の種類（8 文字）
CROTAn	回転角

（n は座標軸の番号、単位は SI 系と角度の「度」）
これにより、CROTAn=0.0 の場合、座標軸 x_n はインデックス n から次式で計算される。

$$x_n = CRVAL_n + CDELT_n \times (n - CRPIX_n) \tag{2.17}$$

これはあまりにも単純であり、もっと一般的な表現方法として WCS が提案された。

2.3.4 World Coordinate System (WCS)

D.wells は 1981 年ころから、天球座標とデータ配列（天体イメージの x,y とかだけでなく、スペクトルの波長軸やストークスパラメータのようなものも含めて）の間の対応を表現するためのシステムが必要であることを認識し、必要なキーワードの提案を行っていた。これが World Coodinate System (WCS)[4] の端緒である。その後、電波天文分野の制約ソフトウェアである AIPS の開発に関連して、Greisen はもう少し詳しい規約を提案し、これらは電波天文分野をはじめ、他の分野（X 線や赤外線など）にも波及していった。

WCS の基本変換

WCS の提案では、ピクセル座標から世界座標への変換は 3 つのステップを踏んで変換される。流れを図で示すと次のようになる。

[ピクセル座標 p_i]
↓ step1 ⋯ 線形変換（行列をかけ、回転、歪みスケールの補正）
↓
[中間ピクセル座標 q_i]
↓ step2 ⋯ 物理単位へ再スケーリング
↓
[中間世界座標 x_i]
↓ step3 ⋯ 座標変換（球面から平面への射影と、実世界座標への変換）
↓
[世界座標 (World Coodinate)]

まず、最初の step1 としてピクセル座標から中間ピクセル座標への線形変換をする。このためにはピクセル座標ベクトル p_j に対して以下の行列計算をする。

$$q_i = \sum_{j=1}^{n} m_{ij}(p_j - r_j) \tag{2.18}$$

ここで、r_j は CRPIXj で与えられる参照点でのピクセル座標であり、m_{ij} が変換行列、q_i が中間ピクセル座標である。これ以降、添字の j はピクセル軸を、i は世界軸を表す。m_{ij} は n × n の正方行列であり N は NAXIS キーワードで与えられる。ただし、この点は WCSAXES キーワードによって一般化される。変換結果の q_i は、中間世界座標軸と一致する方向の中間ピクセル座標軸ベクトルであり、無次元のピクセル単位での参照点からのオフセットである。

次に step2 として、q_i を対応する中間世界座標の x_i に変換する次式のような計算をする。

$$x_i = s_i q_i \tag{2.19}$$

最後に step3 は中間世界座標から世界座標への変換である。具体的には球面から平面への射影法と平面と天球面の接点での世界座標の値から決まる変換により実際の世界座標に変換する。この変換は CTYPEi に依存する。単純な線形軸では、x_i は CRVALi で与えられる参照点における座標軸に加えるオフセットと解釈される。それ以外の場合には、CTYPEi は x_i、CRVALi と他のパラメータの関数を規約に従って定義することになる。規約にない CTYPEi は線形と解釈される。非線形座標は CTYPEi に”4-3”形式で記述される。例えば VOPT-F2W のようなもので、最初の 4 文字が座標の種類を表し、5 番目の文字は’-’ で、残りの 3 文字が中間世界座標から世界座標に変換するアルゴリズムを指定する。座標の種類が 4 文字に満たない場合は’-’ で補い、アルゴリズムが 3 文字に満たない場合は空白を補う。

第3章　赤外線天文衛星「あかり」

3.1　赤外線

人間が視覚することのできる電磁波は、紫外線より長く赤外線より短い 0.4〜0.75μm の間の波長域である。波長が 0.75〜1000μm（1mm）間の電磁波を赤外線といい、1mm 以上の電磁波を電波という。

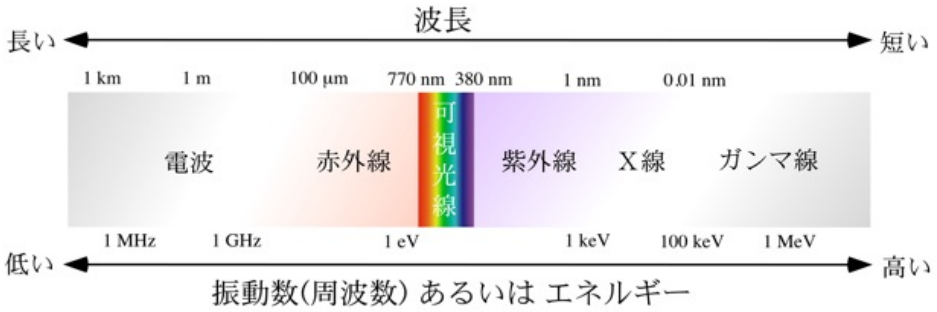

図 3.1: 電磁波スペクトル図[2]

図 3.1 は各電磁波のスペクトル図である。赤外線可視光よりも波長が長く、エネルギーの低い電磁波 [5] であり、波長によって近赤外線、中間赤外線、遠赤外線に分けられる。
また、物質は黒体放射により温度に応じた電磁波を放射し、波長 0.75 ー 1000μm の赤外線は黒体放射の温度 3863 ー 2.89K に対応する。すなわち、赤外線は生命の体温程度の温度を伴う天体現象を観測することができると言える。例えば、宇宙空間の塵や埃など物質の循環について観測ができる。さらに、遠方にある天体を観測する際には、天体からの光は宇宙膨張によるドップラー効果を受け、長波長側に変移する。すなわち、宇宙初期の頃の解析や宇宙の様子を理解することができる。

図 3.2 は私たちの銀河である天の川銀河を、赤外線と可視光で見た違いを表した図である。可視光では暗く映っている部分でも赤外線から見ると明るくなっていることが分かる。このように普段見ることのできない宇宙の様子を観測することができる。

図 3.3 は各電磁波の波長と大気の吸収率のグラフである。赤外線は大気の吸収率が非常に高いため、地球上での観測は困難である。

現在、赤外線での宇宙の様子を観測するため、日本の赤外線天文衛星「あかり」が打ち上げられている。

3.2　赤外線天文衛星あかりの概要

あかり（第 21 号科学衛星 ASTRO-F）[6] は日本の JAXA(宇宙航空研究開発機構) 宇宙科学研究本部を中心として計画が進められてきた赤外線天文衛星である。図 3.4 に「あかり」の外観を示す。

[2]http://www.isas.ac.jp/ISASnews/No.251/intro.html

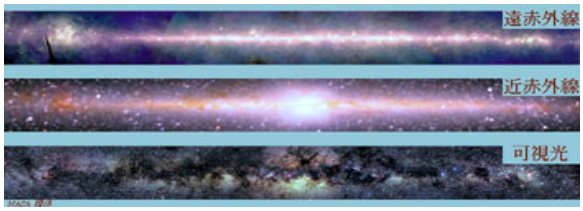

図 3.2: 光による宇宙の見え方の違い　出典：JAXA

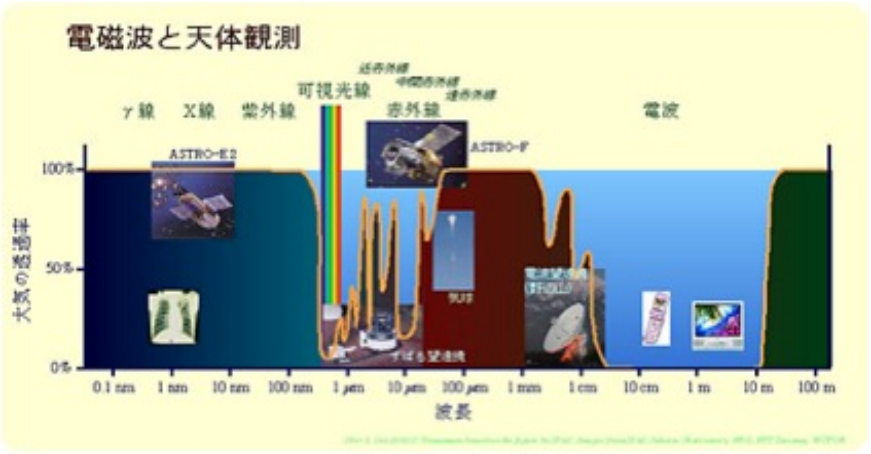

図 3.3: 波長と大気の吸収率のグラフ　出典：宇宙航空研究開発機構（JAXA）

2006 年 2 月 22 日に M-V ロケット 8 号機により打ち上げられた。あかりの主な目的は全天サーベイ観測であり、1983 年にアメリカ、イギリス、オランダが開発、打ち上げを行った IRAS(Infrared Astronomy Satellite) と同様の目的である。IRAS の望遠鏡は有効口径が 57cm であったのに対し、あかりの望遠鏡は有効口径が 68.5cm であるため、IRAS よりも広視野の観測が可能となった。表 3.1 は「あかり」の基本的な特性である。

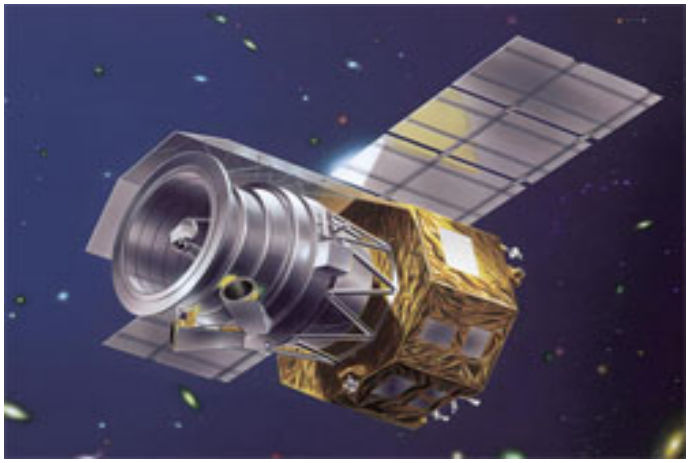

図 3.4: 「あかり」の外観　出典：JAXA

打ち上げ	2006 年 2 月 22 日
全高	3.7 m
幅	5.5 m（太陽パドルの端から端まで）
総重量	952 kg
高度	700 km
軌道周期	100 分
傾斜角	98.2 度

表 3.1: 「あかり」の基本的特性[5]

図 3.5 はあかりの簡単な構造図である。液体ヘリウム及び機械式冷凍機を用いた冷却望遠鏡とそれを維持するクライオスタットからなるミッション部と、姿勢制御やデータ送受信などを受け持ち、衛星を維持するためのバス部から構成されている。焦点面には、遠赤外線を観測する FIS(Far-Infrared Surveyor) と、近赤外線・中間赤外線カメラである IRC(InfraRed Camera) の 2 種類の観測装置が搭載されている。あかりの主要ミッションである全天サーベイ観測は 2006 年 5 月 8 日から行われ、2007 年 8 月 26 日に冷却用の液体ヘリウムを使い切ったため終了した。その期間 550 日に及び、全天の約 94 %の領域について遠赤外線サーベイ観測を、中間赤外線サーベイ観測や 5000 回以上の指向観測を達成した。

[3]http://www.isas.ac.jp/j/enterp/missions/akari/index.shtml

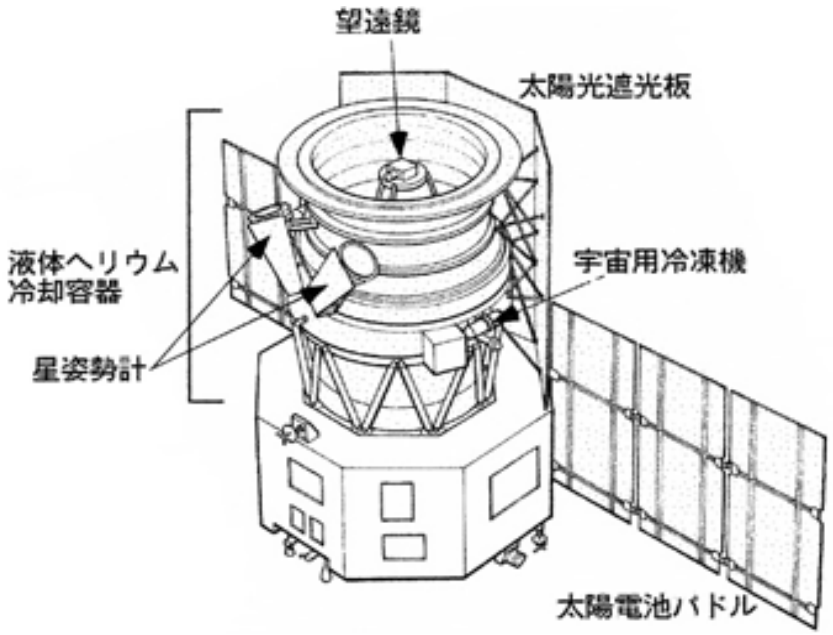

図 3.5: あかりの構造図　出典：宇宙科学研究本部（ISAS）

3.3 観測装置

3.3.1 FIS：Far-Infrared Surveyor（遠赤外線サーベイヤー）

FIS[7] は遠赤外線で全天サーベイを行うことを主な目的として搭載されている観測装置である。表 3.2 は FIS の基本特性である。検出器にはゲルマニウムに少量のガリウムをドープした半導体結晶 (Ge:Ga) を使用している。またこの素子を $1mm^2$ あたり 40〜60kg 重の力で加圧することで圧縮した Ge:Ga 検出器ができる。FIS では Ge:Ga 検出器と圧縮した Ge:Ga 検出器の 2 台を搭載しており、さらにそれぞれをフィルターで区切って使用するため、4 つのバンドでの観測を行うことができる。

バンド名	N60	WIDE-S	WIDE-L	N160
観測波長 (μm)	50-80	60-110	110-180	140-180
検出素子	Ge:Ga		圧縮型 Ge:Ga	
ピクセル数	20 × 2	20 × 3	15 × 3	15 × 2
ピクセルサイズ (秒角)	26.79		44.20	
ピクセルピッチ (秒角)	29.47		49.11	
読み出し回路	Capacitive Trans-Impedance Amplifire (CTIA)			
サンプリング間隔 (Hz)	25.28		16.86	

表 3.2: FIS の基本特性　出典：JAXA

3.3.2 IRC：Infrared Camera（近赤外線・中間赤外線カメラ）

IRC[8] は 3 台の独立したカメラシステムから構成されている。表 3.3 に IRC の特性を示す。1.7-5.5μm の近赤外線領域を NIR カメラ、5.8-14.1μm の中間赤外線領域の短波長側を MIR-S カメラ、12.4-26.5μm の中間赤外線領域の長波長側を MIR-L カメラが担当している。

カメラ	NIR	MIR-S	MIR-L
検出器の種類	InSb	Si:As	Si:As
ピクセル数	512 × 512	256 × 256	256 × 256
撮像領域	9.5 × 10.0	9.1 × 10.0	10.3 × 10.2
ピクセル視野（秒角）	1.46	2.34	2.51 × 2.39

表 3.3: IRC の基本特性　出典：JAXA

特徴の一つは、それぞれに大規模アレイ (512 × 512 素子、256 × 256 素子) を採用したことによって、10 分平方角という広視野を一度に観測できることである。それぞれのカメラではフィルターホイールを回転させることによって観測対象の波長帯をより細かく選ぶことができる。IRC での観測は指向観測モードで実施されるほか、検出器の一部を使用して FIS と同様サーベイ観測を行うことも検討されている。

3.3.3 望遠鏡

あかりの望遠鏡はF/6.1のリッチークレチアン (Ritchey-Chretien) 式の光学系である。図3.6は主鏡の構造である。望遠鏡の焦点距離は4200mmで、主鏡の有効口径は68.5cm。主鏡、副鏡のほか、副鏡を支えるトラス (truss) や迷光を防止するバッフル (buffle) などから構成されている。主鏡の材質は軽量で丈夫なシリコン・カーバイト (SiC) で、さらに軽くするために裏面が大きくくり抜かれた構造になっている。これにより、直径71cmであるにも関わらず重さが約11kgという軽量化が実現された。

図 3.6: 主鏡の構造　出典：JAXA

3.4 あかりの観測

3.4.1 衛星の姿勢モード

あかりの観測はサーベイモードとポインティングモード（指向観測モード）の2種類の衛星姿勢モードがある。図3.7、3.8が各姿勢モードの状態図である。

サーベイモードはあかりの基本観測モードで、望遠鏡の向きが太陽とも地球の中心とも常に垂直になるように制御されている。衛星が地球の周りを一周するごとに望遠鏡も天球上を一周観測している。地球から見た太陽の方向は1年で一周することから半年間で全天の観測をすることができる。

ポインティングモードでは天球上のある方向を長時間積分観測、あるいは分光観測を行いたい場合に使用される。ただ、望遠鏡を低温に保つため太陽や地球の光が入射する方向に望遠鏡は向けることができない条件があるため自由に方向を変えることはできない。

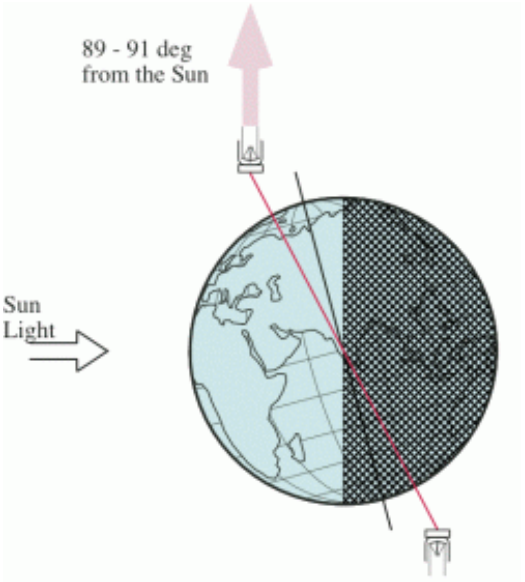

図 3.7: サーベイモード

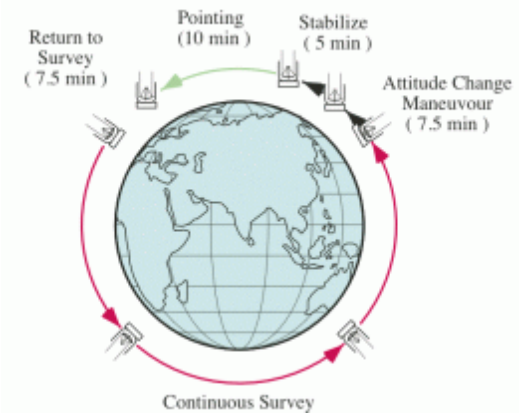

図 3.8: ポインティングモード　出典：JAXA

3.4.2　全天サーベイ観測

「あかり」は全天サーベイの成果として赤外線を使った 20 年ぶりの天体カタログ、すなわち宇宙の地図を作り直すことを目的としている。1983 年に世界初の赤外線天文衛星「IRAS」が全天サーベイを行った。IRAS の観測によって得られた赤外線天体のデータベース（宇宙地図）は、その後の可視光や電波、X 線での観測を後押しして天文学の進歩に非常に大きな貢献をしたのだが、20 年がたって最先端の天文学の要求に合わなくなってしまった。現在の天体観測でも 20 年前の地図が使用され、暗い天体や細かい部分ははっきり見えていない。そこで最新技術のを投入し「あかり」での赤外線観測を行うことになった。衛星の観測姿勢はサーベイモードで行っている。

「あかり」の赤外線天体カタログは、波長 9、18μm の中間赤外線で検出した約 70 万天体のカタログと、65、90、140、160μm の遠赤外線 4 波長の情報を持つ約 6 万 3000 天体のカタログの二つのパートからなる。図 3.9 は波長 9μm の中間赤外線全天サーベイの様子。この画像は共同研究者である海老沢教授（宇宙航空研究開発機構：JAXA, 同宇宙科学研究本部：ISAS）が「あかり」チームに作成を依頼した。中心から帯状に左右に広がる明るい部分は、天の川銀河の円盤部分をその中にいる地球から真横に見たもので、我々の銀河系の中心の方向にあたる。この方向は、塵や年老いた赤く、明るい星（赤色巨星）が密集していて特に明るく見える。

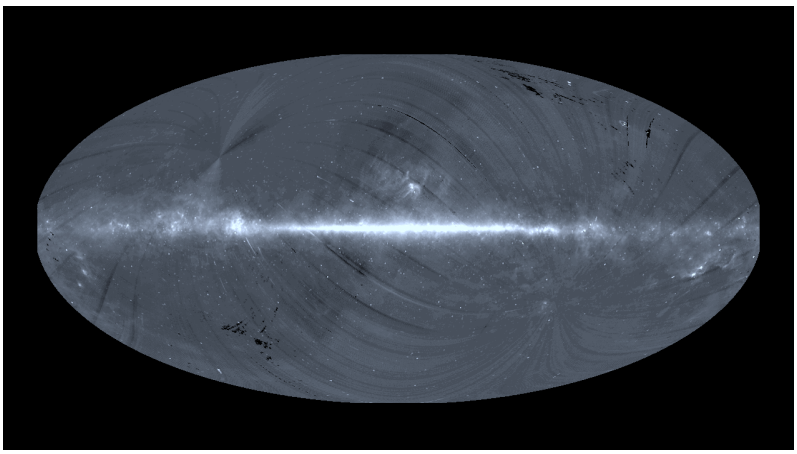

図 3.9: 9μm での「あかり」全天サーベイ　出典：JAXA「あかり」チーム

第4章　「あかり」データのフォーマット変換

本総合研究の開発項目として、§3.4.2 の図 3.9 で紹介した「あかり」全天画像のデータを用いてデジタルプラネタリウム投影用のフォーマットへの変換を試みる。しかしただ変換といっても、デジタルプラネタリウムへ投影するための投影法には決まりがあり、変換の際に、入力 FITS ファイルと出力 FITS ファイルの間に大幅な解像度の違いがあった場合は、プログラムの動作に影響があることも問題となる。ここでは、開発の準備において昨年度の研究成果に基づいた方法を応用しながら、様々な衛星観測データに対応できるフォーマット変換方法を探していく。

4.1　投影用フォーマットの決定

デジタルプラネタリウムでは、全天画像もデジタル画像として扱うため、作成する画像にデジタル処理があまり影響を与えないような投影法が好ましいとされる。§2.2.3 にていくつか紹介した投影法の中で、TSC(Tangential Spherical Cube) 投影法は 1 つ 1 つの分割に対して歪みが少なく捉えられる。従ってデジタル画像に対しても歪みの影響が少ないというところで、TSC を採用する。

また、解像度にも決まりがある。解像度とは、1 つの画像に対して格子状に分けられた点（画素）がどのくらい細かく敷き詰められているかを表す数値である。今回デジタルプラネタリウムの仕様としては縦横の規格が 2 の階乗（1024, 2048, 4096 など）が好ましいとされている。TSC は組み立てると立方体の形になるため、それが都合の良い規格である。。TSC の一面が 2048×2048pixel、6 面全体で 8192×6144pixel の解像度とする。

4.2　座標変換サブルーチンの作成

次に昨年度の ROSAT データ用に使われたプログラムを使用して「あかり」データを走らせる。しかし、「あかり」FITS ファイルのヘッダ部には書き込まれている座標は銀河座標となっている。§2.2.1 で説明したようにプラネタリウムでは我々が宇宙の様子を見やすいように、赤道座標で表示されている。よって銀河を中心として表示される銀河座標では都合が悪い。
プログラム上で天球座標とデータ配列（画素値）を対応付けている WCS は、

| 入力画像 FITS ファイル |
↓
↓　　← ヘッダの座標が共に赤道座標ならば WCS が動作する
↓
| 出力画像 FITS ファイル |

という関係をなしているため、銀河座標 ↔ 赤道座標の変換を行わない。プログラム内に銀河座標から赤道座標へ変換するルーチンを作成する。入力画像 FITS のヘッダ部の座標を読み出して銀河

座標であれば、座標変換サブルーチンへ移動し、赤道座標の座標値に直してから出力画像のピクセル値へと算出していく（WCS の動作）。銀河座標 → 赤道座標の計算式は §2.2.2 の式を引用した。

4.3 西尾（2007）の手法による変換と問題点

4.3.1 ROSAT 画像による昨年度の結果の再現

変換において必要なものは、

- 入力 FITS ファイル：天文衛星が観測した画像データ
- 変換用のプログラム

である。昨年度の研究成果として、2007 年度西尾光史総合研究論文 [9] で、X 線天文衛星 ROSAT の観測データが既に提供されていたプログラムによって、デジタルプラネタリウム用フォーマットへ変換されている。ROSAT 全天データは 1378 枚に分割されており、3 つのエネルギー帯域で観測されている。それらを一枚ずつ張り合わせたものから投影用データを作成している。図 4.1 は昨年度プログラムの流れをフローチャートで示したものである。

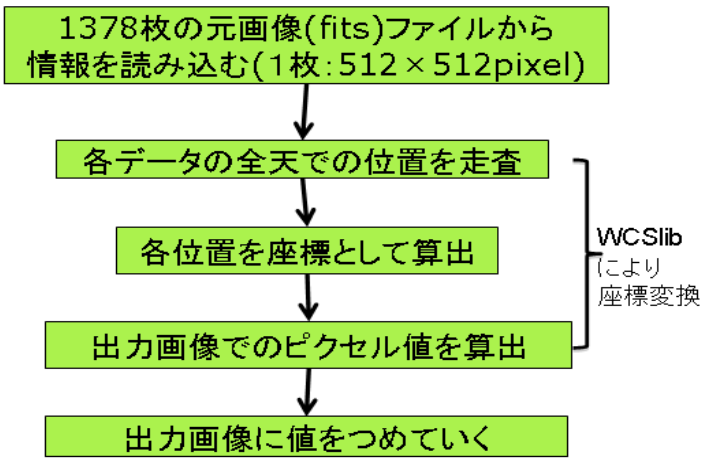

図 4.1: ROSAT プログラムフローチャート

このプログラムの動作としては、まず一枚ずつの入力画像を読み取る。ここでは、CFITSIO[4] という FITS 入出力用ライブラリによって読み込まれている。次に各入力 fits ファイルのデータのピクセルに対応する天球座標を算出し、出力画像のピクセルを算出している。デジタル画像処理のピクセル値の対応は入力→出力である。この入力画像からそのまま出力画像へ変換する方法を順変換という。その動作を 1378 枚の画像全てに行い、張り合わせて出力画像を取得している。動作を確認するため、昨年度の取り組みを実際に行った。図 4.2 が昨年度のプログラムで取得した画像である。

図 4.2 は ROSAT データから作成した全天画像である。3 つのエネルギーバンドでそれぞれ作成し、STIFF[10] というコマンドで疑似カラー合成をしている。X 線だから宇宙がこういった色で見えているわけではなく、3 色に分けて表示することで宇宙のエネルギーの分布などを知ることができる。例えば、青く帯状の部分は我々の天の川銀河を示していて、エネルギーが高いことがわかる。「あかり」の捉えた赤外線での様子もまた違ったものになるだろう。

[4] http://heasarc.gsfc.nasa.gov/fitsio/

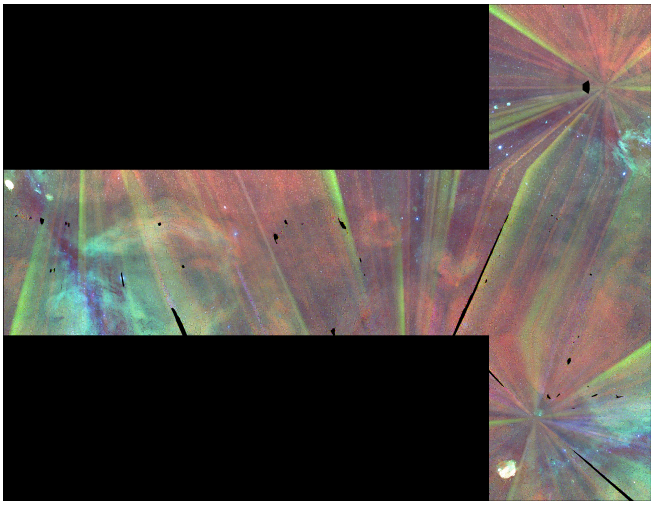

図 4.2: 昨年度の取り組み（ROSAT 全天画像）

4.3.2 「あかり」全天画像の変換と問題点

変換コマンド

作成した座標変換の関数を加え、「あかり」用プログラム作成した。変換時に以下のコマンドを入力する。

% ./convertWCS TSC 8192 6144 7168.5 3072.5 0.0439453125 AllSky_S_p5.fits.gz Akari_p5_TSC.fits

これは左から、走らせるプログラムのソースファイル、投影法、解像度（横）、解像度（縦）、参照ピクセル座標（横）、参照ピクセル座標（縦）、座標スケール、入力 fits ファイル名、出力 fits ファイル名の順になっている。
これは投影法によって違い、上記のコマンドは TSC 投影法で作成する際の入力値である。

投影法	解像度	参照ピクセル座標	座標スケール
TSC	8192×6144	7168.5, 3072.5	0.0439453125
CSC	8192×6144	7168.5, 3072.5	0.0439453125
CAR	4096×2048	2048.5, 1024.5	0.087890625

表 4.1: 各投影法の入力パラメータ

表 4.1 は投影法別のコマンド入力時のパラメータである。TSC と CSC はほぼ同じ形であるため、パラメータに違いはないが、CAR の場合は §2.2.3 のように長方形であるため別の入力値となる。

コマンドを入力しプログラムを走らせた。

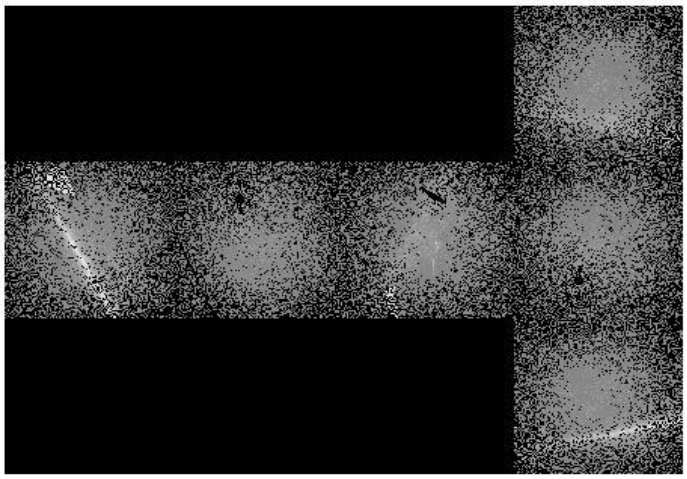

図 4.3: 作成した「あかり」全天画像

図 4.3 は作成した「あかり」全天画像である。図 4.2 と比較したところ、確かに TSC 投影法の形に変換されていることは分かるが、図 3.9 で示したような入力画像にはない隙間だらけの画像となってしまった。これには入力画像と出力画像の間に解像度の影響があると考えた。

問題点

今回、デジタルプラネタリウム投影用の解像度は §4.1 で決めたように、8192×6144pixel である。このファイルサイズを計算すると、
8192×6144=50331648
約 50M 程度の画像となる。
しかし、元の「あかり」入力ファイルのサイズは 6800×3600 である。計算すると、
6800×3600=24480000
約 24M 程度の画像となる。つまり入力ファイルのサイズに対して、出力ファイルのサイズが大きいために画素が不足してしまい、正しい変換ができていないことが分かった。

また、昨年度 ROSAT データが図 4.2 のように正しく変換できているように見えるのは、1 枚のファイルサイズが 512×512 であり、それが 1378 枚ある。計算すると、
512×512×1378=361234432
約 361M 程度の画像であり、十分に画素が満たされていたためである。
画素の不足分を埋めるには、デジタル画像処理での補間 [11] という方法を用いる必要がある。

4.4 逆変換手法の導入

デジタル画像の変換には 2 つ方法があり、1 つは §4.3.1 で述べたように昨年度使用されたような入力画像をそのまま出力画像へ変換する順変換である。しかし先であがった問題のように、入力画像の解像度が出力時の解像度より劣っている場合、１つ１つのピクセルの回転や拡大縮小などの幾何学変換をした際の画素値の抜けた部分に何も処理がされず、正しい変換とはならない。

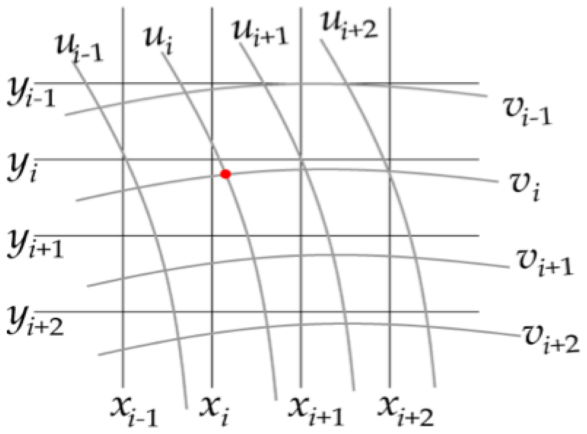

図 4.4: デジタル画像の幾何学変換[5]

図 4.4 はデジタル画像の幾何学変換の様子を表したものだが、デジタル画像は回転や拡大処理を行っただけで整列していた画素の位置が簡単にずれてしまう。そのずれた部分には画素値が存在せず、結果穴の開いた画像が作成されてしまう。そこでもう一つの方法として逆変換を適用することにした。これは先ほどの順変換とは反対に、出力画像の位置から画素の抜ける部分を入力画像から探し出してピクセル値を求める方法である。図 4.5 は逆変換を行うためのフローチャートである。出力画像から入力画像を走査し、画素が一致すればその位置の画素をそのまま適用し、違っていれば補間 (§5.1 参照) という方法で画素の穴埋めをする。

図 4.5 のフローチャートを参考に、使用していた ROSAT データ用プログラムを変更した。WCS が動作している部分は常に入力画像の値を先行して処理を行っている。これを出力から走査を始めるように書き換え、さらに §4.2 で示した座標変換のサブルーチンも追加した。入力が銀河座標ならば、出力を銀河座標へ変換してからピクセル値の算出を行う。

再度プログラムを走らせ、「あかり」全天画像を作成した。

図 4.6 は改善後の「あかり」全天画像である。逆変換から補間により画像の穴抜けの部分が満たされていることがわかる。ここで §5.1.1 で述べるニアレストネイバーという補間法を用いて作成している。図 4.2 の ROSAT 全天画像と比べると宇宙の姿はまた別の見え方をしていることがわかる。先ほど天の川銀河と紹介した帯状の部分は X 線では暗く、見えていなかったが、赤外線では明るく見えている。これは赤外線が X 線に吸収されていたために X 線観測装置では映らなかったためである。以上より逆変換の導入に成功した。また、§2.2.3 で紹介したように各投影法での全天画像も作成した（図 4.7、4.8)。

[5]http://sango.lab.tkikuchi.net/Members/tkikuchi/courses/img2007/07/

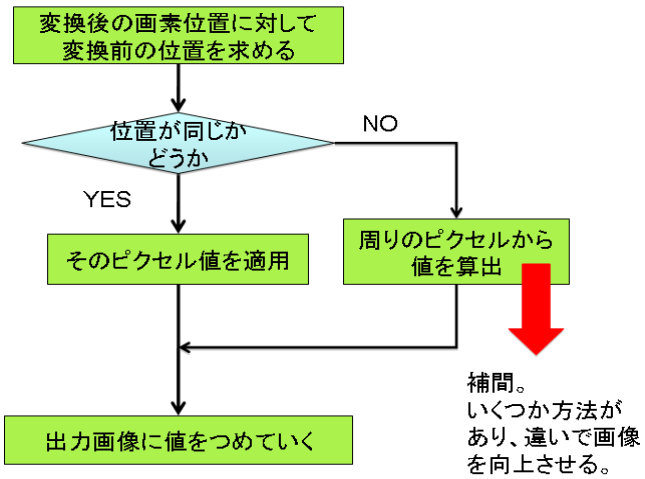

図 4.5: 逆変換のフローチャート

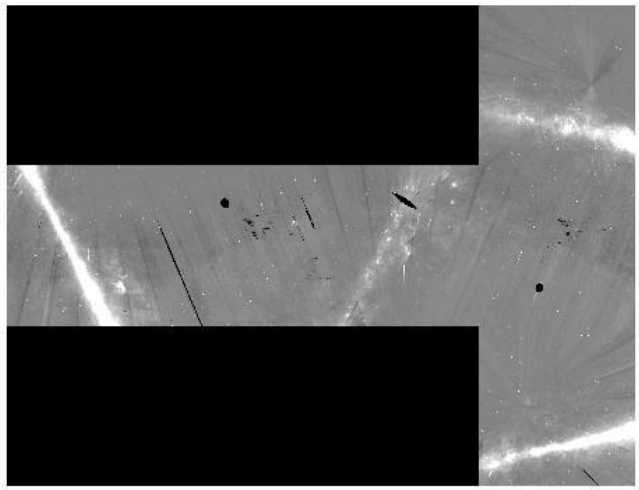

図 4.6: 改善後の「あかり」全天画像

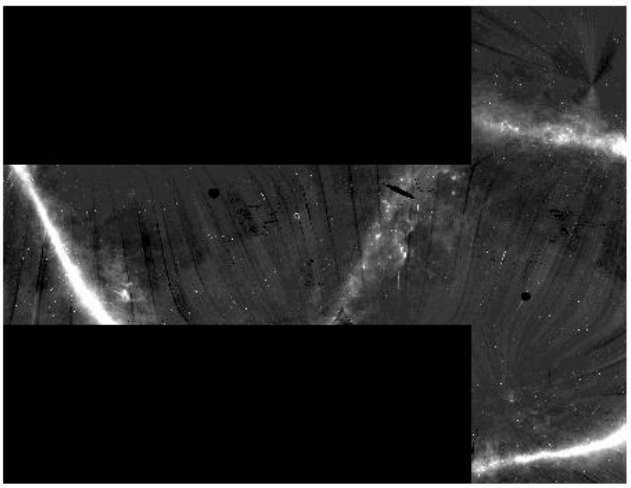

図 4.7: CSC 投影法

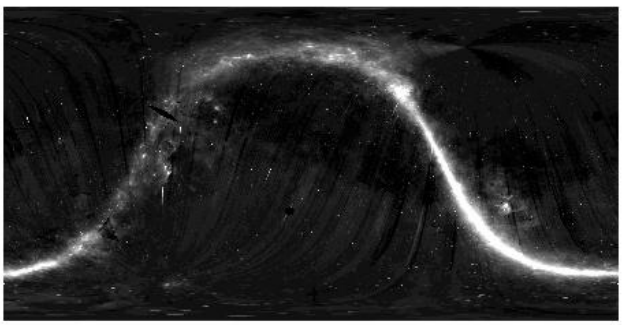

図 4.8: CAR 投影法

第5章　補間法の検証

5.1　補間法

§4.4 で示したように、逆変換において画素値の足りない部分は補間によって埋めることができる。本研究で活用した補間法は 3 種類である。それぞれに特徴があり、参照とする画素の数が多い程、得られる画像の質は良いと予想される。それぞれ作成した画像の質について最適な方法を検討していく。

5.1.1　ニアレストネイバー法

補間法として最も単純なものは、図 5.1 に示すように、求めたい位置に最も近い画素位置の値をそのまま利用するニアレストネイバー (nearest neighbor) という方法である。

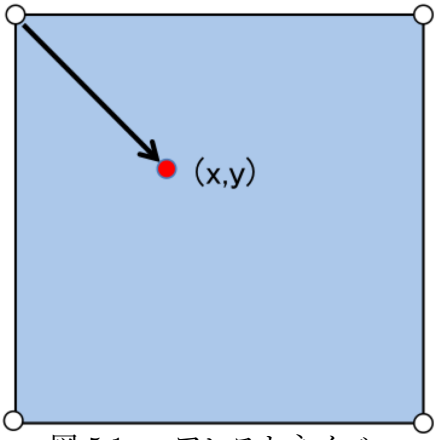

図 5.1: ニアレストネイバー

値を求めたい位置を (x,y) とすると、その位置の画素値 I(x,y) は次式で表される。

$$I(x, y) = f([x + 0.5], [y + 0.5]) \tag{5.1}$$

ただし、f(i,j) は入力画像の位置 (i,j) の画素値を、記号 [ ] はガウス記号である。
この方法は処理が非常に単純で計算時間も高速であるが、本来滑らかなエッジがギザギザになって現れるジャギーが発生しやすい。

5.1.2　バイリニア補間

図 5.4 に示すように、求めたい位置 (x,y) の値 I(x,y) を、周りの 4 点の画素値を用いて求める方法をバイリニア (bilinear interpolation) 補間という。

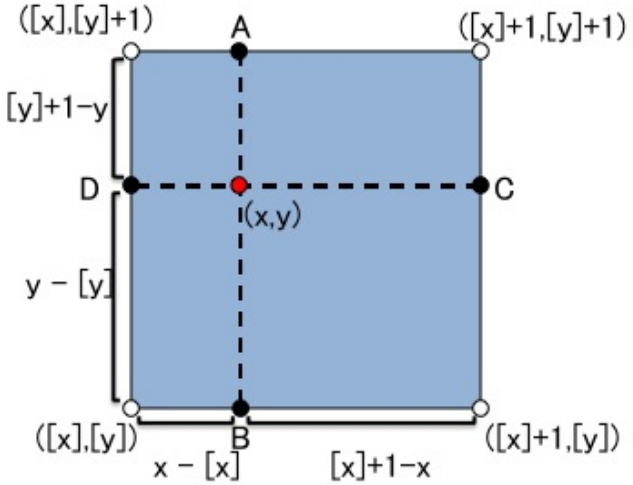

図 5.2: バイリニア補間

計算式は次のようになる。

$$I(x,y) = \begin{pmatrix} [x]+1-x & x-[x] \end{pmatrix} \begin{pmatrix} f([x],[y]) & f([x],[y]+1) \\ f([x]+1,[y]) & f([x]+1,[y]+1) \end{pmatrix} \begin{pmatrix} [y]+1-y \\ y-[y] \end{pmatrix} \tag{5.2}$$

この計算は、同図の A 点と B 点の値を、まずそれぞれの左右値をもちいて線形補間することから求め、さらに求めた A、B 点の値を縦方向に線形補間することにより、位置 (x,y) の値 I(x,y) を求めることに等しい。あるいは、上下方向から先に線形補間し、後に横方向の線形補間を行っても同じ結果が得られる。
バイリニア補間では、周り 4 点の画素値の重み付けの平均値を求めることになるため、平滑化の効果が生じる。平滑化は簡単にいうと「ぼかし」のことである。そのため、ニアレストネイバーのようなジャギーが目立たなくなるが、一方でエッジがなまってしまう傾向がある。

5.1.3 バイキュービック補間

バイキュービック補間 (bicubic interpolation) では、図 4.5 に示すように求めたい位置 (x,y) の値 I(x,y) を周りの 16 点の画素値 $f_{11}, f_{12}, \cdots, f_{44}$ を用いて、次式により求める。

$$I(x,y) = \begin{pmatrix} h(x_1) & h(x_2) & h(x_3) & h(x_4) \end{pmatrix} \begin{pmatrix} f_{11} & f_{12} & f_{13} & f_{14} \\ f_{21} & f_{22} & f_{23} & f_{24} \\ f_{31} & f_{32} & f_{33} & f_{34} \\ f_{41} & f_{42} & f_{43} & f_{44} \end{pmatrix} \begin{pmatrix} h(y_1) \\ h(y_2) \\ h(y_3) \\ h(y_4) \end{pmatrix} \tag{5.3}$$

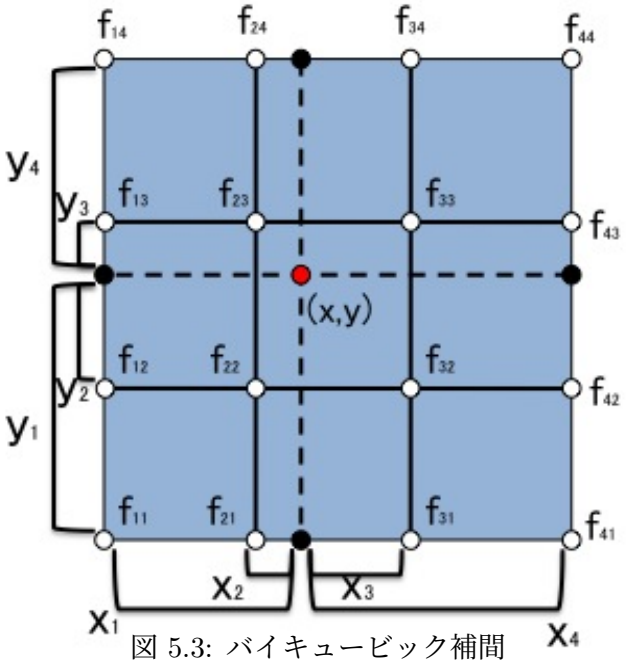

図 5.3: バイキュービック補間

ただし、

$$
\begin{aligned}
x_1 &= 1 + x - [x] \\
x_2 &= x - [x] \\
x_3 &= [x] + 1 - x \\
x_4 &= [x] + 2 - x \\
y_1 &= 1 + y - [y] \\
y_2 &= y - [y] \\
y_3 &= [y] + 1 - y \\
y_4 &= [y] + 2 - y
\end{aligned}
\tag{5.4}
$$

この方法ではバイリニア補間に比べて、よりシャープで自然な画像が得られる。しかし計算時間は 3 種類の内最も長い。また、関数 h(t) は sinc 関数 ((sinc=sin(πt)/πt)) を 3 次多項式で近似するもので、一般に次の式が用いられる。

$$
h(t) = \begin{cases} |t|^3 - 2|t|^2 + 1 & (|t| \leqq 1) \\ -|t|^3 + 5|t|^2 - 8|t| + 4 & (1 < |t| \leqq 2) \\ 0 & (2 < |t|) \end{cases}
\tag{5.5}
$$

5.2 「あかり」全天画像への補間法の導入

正しく変換された全天画像を取得することはできたが、実際にプラネタリウムへ投影する際にはここで示した画像は大きく映る。デジタルで作成されている画像はどの程度の精度で表されているのかを人間の目の分解能を考慮して考察していく。

5.2.1 各補間法の比較

§5.1 で説明したように 3 種類の補間法をそれぞれ変換プログラムに差し替えて、3 枚の全天画像を作成し、天体などの画像がどの程度の大きさであるかを確認した。図 5.4 は全天における天の川銀河付近のガス状のものを拡大して見たものである。すると、各補間法での効果の違いがはっきり伺える。ニアレストネイバーではジャギーが発生してしており、プラネタリウムの画像としては荒く映るであろうことがわかる。そして、バイリニア補間、バイキュービック補間の順に画像が滑らかになっていることがわかる。バイキュービック補間法に注目して見てみると、大きさは約 5 分角のものであった。

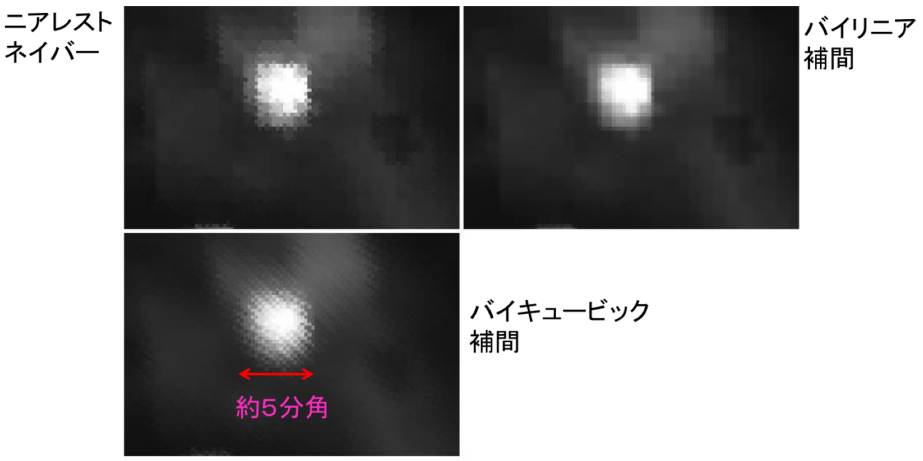

図 5.4: 各補間法の比較

5.2.2 考察 ー最適な補間法ー

次に、人間の目では実際にどの程度の画像を分解できるかを、体感を通して考察する。これを行うことで今後のデジタルプラネタリウム投影における画像の質に関して検討していく。

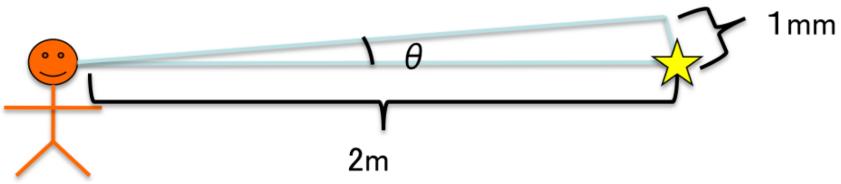

図 5.5: 人間の目の解像度

図 5.5 は人間の目で見る対象物の距離と大きさを示した図である。2m 離れたところから 1mm 程度の対象物が見える。これを人間の目で分解できる限界だと判断した。なお、人間の目は視力など個人差があるため全ての事柄にあてはまるわけではない。今回は本研究の考察項目としてこの数値をあげることとする。以上の数値から分解能を計算した。

$$\theta = 360\ ^\circ \times \frac{1 \times 10^{-3}}{2\pi \times 2} = 28.64 \times 10^{-3}\ ^\circ = 1.72' \tag{5.6}$$

となる。よって人間の目の分解能は 1.7 分角程度と決定する。
先ほど図 5.4 の対象物が約 5 分角であったことから、この分解能は画質の違いを十分に判断できる数値である。したがって、人間の目では 1.7 分角程度の分解能であれば、§5.1 で示した 3 種類の補間法の違いをプラネタリウム投影において区別できるということが分かる。さらに各補間法を検討すると、バイキュービック補間法が最も良い補間法であるといえる。

5.3 プレビューツール”All Sky Viewer” を用いたバーチャルシミュレーション

作成したコンテンツが実際のプラネタリウムでの投影においてどのように体感できるかをシミュレートするために、All Sky Viewer を用いたバーチャルシミュレーションを行った。

All Sky Viewer はサイト：http://orihalcon.jp/allskyviewer/でダウンロードすることができる。動作環境は現在 Windows のみであるが、個人で作成した画像や映像を実際にシミュレーションすることができる。画像を投影する場合は、TSC で作成した全天画像を 6 枚の正方形の画像に分割してからドームへの張りつけを行う。6 枚の分割には特に方法は決められていないので、画像編集ツールの photoshop を利用した。

作成した「あかり」データの全天画像を All Sky Viewer を使用してシミュレーションを行った。

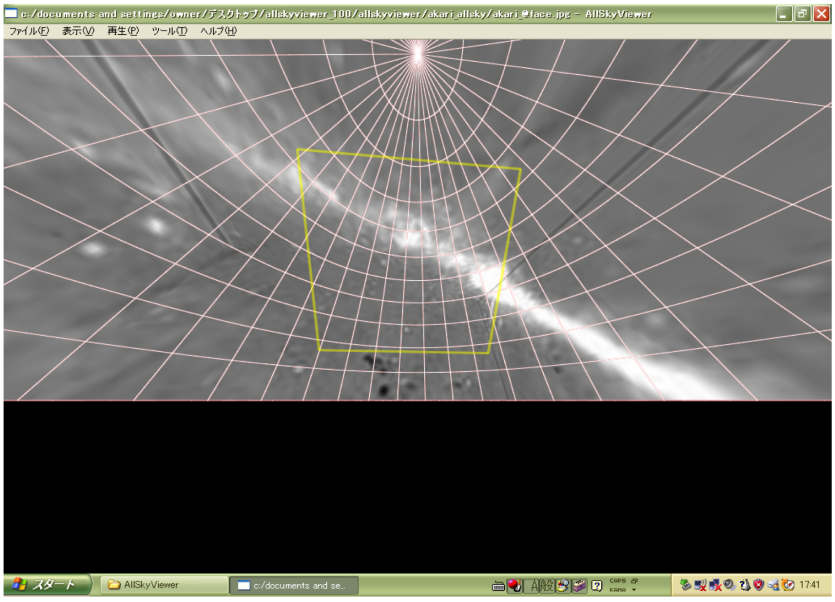

図 5.6: All Sky Viewer を利用した「あかり」全天画像（カメラ視点）

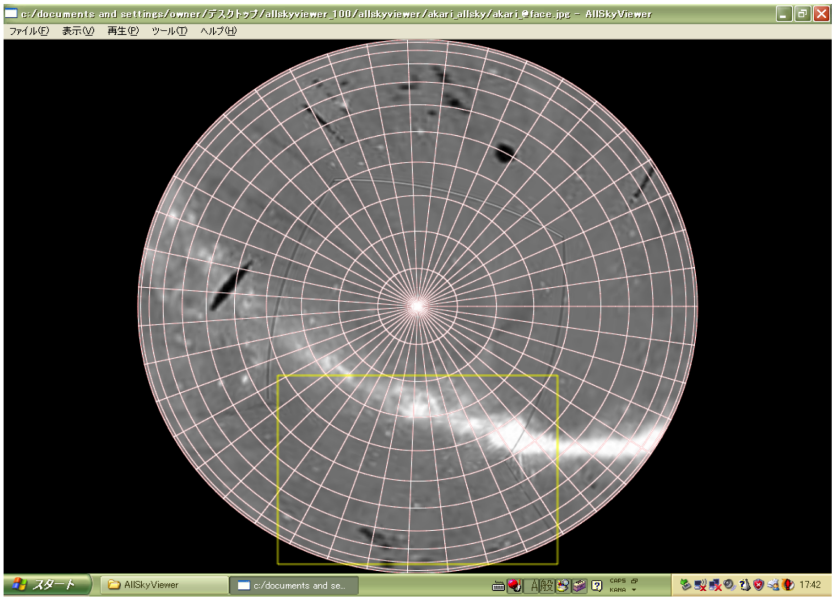

図 5.7: All Sky Viewer を利用した「あかり」全天画像（ドームマスター）

図 5.6、5.7 は「あかり」全天画像を All Sky Viewer で投影シミュレーションした図である。カメラ視点状態とドームマスター状態の 2 つの状態を映し出すことができる。図 5.6 がカメラ視点であるが、全天の様子をプラネタリウムのように内側から眺めているように観察ができる。図 5.7 はドームマスター状態である。プラネタリウムを上から見下ろした視点となっており、実際のプラネタリウムとは視点の違う宇宙の観測もできそうである。

5.4 考察

補間法を利用した逆変換から、「あかり」全天画像の正しい変換ができた。そして分解能の考察から最適な補間法を決定することができた。しかし、補間法においては計算時間の問題もあり、まだ十分な変換ツールとはいえない。

補間法	計算時間	画像の質
ニアレストネイバー	3 分程度	ジャギーが目立つ
バイリニア補間	5 分程度	平滑な画像
バイキュービック補間	10 分程度	平滑化とエッジ強調により画像が鮮明

表 5.1: 「あかり」プログラムにおける比較

表 5.1 は「あかり」データ用変換プログラムにおける、計算時間と画質についてまとめたものである。バイキュービック補間法で画像の質の向上が図れたものの、計算時間は最も長く、改良の余地はまだあるといえる。また、使用した「あかり」全天データには、今回紹介した 1 枚なりのデータと、もう一つ 6 枚に分割されているファイルが提供されている。これらは画像ファイル自体に問題があり、本研究でのプログラムで動作しないことがわかっている。様々な衛星観測データに対応できるように、デジタルプラネタリウム投影用変換プログラムへ改良することが今後の課題である。

第6章　まとめ

本総合研究では、昨年度の研究成果であったX線天文衛星ROSATの観測データをデジタルプラネタリウム投影用のフォーマットへ変換する取り組みを基に、赤外線天文衛星「あかり」の観測データを活用することへ発展した。観測データはそれぞれFITSというファイル形式で統一し、保存されているものの、画像自体の規格に統一性はない。昨年度ROSAT用に使われたプログラムでは、WCSによって走査された入力画像の座標値をそのまま出力画像のピクセル値へ算出する順変換という方法が採用されていた。しかし、ファイルサイズに統一性のない各衛星観測データは、デジタルプラネタリウム投影用フォーマットへ変換する際に使用される入力画像の解像度によって、出力時のデジタル画像が正しくない場合があることがわかった。特に今回使用した「あかり」画像データの解像度は出力画像の解像度よりも低いために問題が明確にされた。

そこで、デジタル画像処理の基本でもある補間を利用した逆変換という方法を採用することで、画素の少なさを解消し、正しい変換をもたらすことができた。さらに、プラネタリウム投影には全天の細かな部分の見やすさも考慮して、いくつかある補間法を試みた。その結果バイキュービック補間法は人間の目の分解能において十分区別できる最も良い方法であると考察した。この成果は「あかり」だけではなく今後の様々な波長帯域の電磁波の衛星観測データへも応用していく狙いである。

謝辞

本研究を行うにあたって、研究についてのご指導、情報提供をして下さった宇宙科学研究開発機構・宇宙科学研究本部の海老沢研先生とに心から深く感謝いたします。卒業論文作成について様々なご協力、大変お世話になりました。ありがとうございました。そして指導教員の久保田あや先生、研究の基礎から論文など様々なご指導ありがとうございました。また、同期の川辺くん、斉藤くん、砂川くん、田丸さん、別当屋敷くん、みなさんと同じ研究室で過ごせたことができ良かったと感じています。

皆様のおかげで卒業論文を完成させることができたと思います。本当にありがとうございました。

付 録A

A.1　座標変換についての詳細

§2.2.1 で説明した。赤道座標 ↔ 銀河座標の変換についてはベクトルを行列表現から解いていったものである。以下のような順をたどって座標変換の式が導けた。

A.1.1　方向ベクトル

以下は天球上で任意の点の座標であるが、同じ位置を表している。

（赤経、赤緯）＝（281. °00, -4. °07）
（銀経、銀緯）＝（28. °463, -0. °204）

どのように座標変換を行っているのか、以下の計算をたどる。
赤経赤緯を通常、(α, δ）で表す。仮に天球の半径を 1 としたとき、長さが 1 で、(α, δ）の方向ベクトル[6] p を求める。春分点の方向を x 軸、赤道面上の赤経 90 度を y 軸、北極を z 軸、とする右手系を考えると方向ベクトルの xyz 座標は以下のようになる。

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \cos\delta\cos\alpha \\ \cos\delta\sin\alpha \\ \sin\delta \end{pmatrix} \tag{A.1}$$

x^2+y^2+z^2=1 であることから、定義した x,y,z 軸の基底ベクトルをそれぞれ、e_x,e_y,e_z とすると、

$$p = xe_x + ye_y + ze_z = \begin{pmatrix} e_x & e_y & e_z \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{A.2}$$

である。
銀河座標に基づいた、x' 軸、y' 軸、z' 軸と基底ベクトル e'_x、e'_y、e'_z とする。x' 軸は銀河中心を向いていて、$x'y'$ 平面は銀河面と一致している。
方向ベクトル p は銀河座標でも以下のように表すことができる。

$$p = \begin{pmatrix} e_x & e_y & e_z \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} e'_x & e'_y & e'_z \end{pmatrix} \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} \tag{A.3}$$

[6] http://plain.isas.jaxa.jp/ ebisawa/TEACHING/2007Komaba/2007Komaba/node6.html

従って、式 (2.16) と同じ関係が銀河座標と (x', y', z') の間に成立する。ベクトル p の銀河座標を (l, b) とすると、

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} \cos b \cos l \\ \cos b \sin l \\ \sin b \end{pmatrix} \tag{A.4}$$

これを逆に解いて、

$$\begin{pmatrix} l \\ b \end{pmatrix} = \begin{pmatrix} \tan^{-1}(x'/y') \\ \tan^{-1}(z'/\sqrt{x'^2 + y'^2}) \end{pmatrix} \tag{A.5}$$

となり、銀河座標 (l,b) を求めることができる。
以上より、ある天体の赤道座標を銀河座標に変換するには、式によって赤道座標での方向ベクトルの 3 成分 (z, y, z) を求め、それを式によって銀河座標の 3 成分 (x', y', z') に変換し、さらに式を用いればよい。

A.1.2 直行行列と変換行列

基底ベクトル（e_1, e_2, e_3）で表される直行座標系[7] と（e'_1, e'_2, e'_3）で表される直行座標系の間の直行座標系を考える（添字 1,2,3 が上で書いた x,y,z に対応）と、これらはそれぞれ互いに単位ベクトルの組であるから、

$$e_1{}^2 = e_2{}^2 = e_3{}^2 = 1, e_1 \cdot e_2 = e_2 \cdot e_3 = e_1 \cdot e_3 = 0 \tag{A.6}$$

$$e'_1{}^2 = e'_2{}^2 = e'_3{}^2 = 1, e'_1 \cdot e'_2 = e'_2 \cdot e'_3 = e'_1 \cdot e'_3 = 0 \tag{A.7}$$

片方の系のベクトルはもう片方のベクトルをつかって表すことができる。

$$e'_1 = a_{11}e_1 + a_{12}e_2 + a_{13}e_3 \tag{A.8}$$

$$e'_2 = a_{21}e_1 + a_{22}e_2 + a_{23}e_3 \tag{A.9}$$

$$e'_3 = a_{31}e_1 + a_{32}e_2 + a_{33}e_3 \tag{A.10}$$

行列表示すると、

$$\begin{pmatrix} e'_1 & e'_2 & e'_3 \end{pmatrix} = \begin{pmatrix} e_1 & e_2 & e_3 \end{pmatrix} \begin{pmatrix} a_{11} & a_{21} & a_{31} \\ a_{12} & a_{22} & a_{32} \\ a_{13} & a_{23} & a_{33} \end{pmatrix} \tag{A.11}$$

4.10 の条件式より、この変換行列の各要素の間に、

$$a_{11}{}^2 + a_{12}{}^2 + a_{13}{}^2 = 1, a_{21}{}^2 + a_{22}{}^2 + a_{23}{}^2 = 1, a_{31}{}^2 + a_{32}{}^2 + a_{33}{}^2 = 1 \tag{A.12}$$

$$a_{11}a_{21} + a_{12}a_{22} + a_{13}a_{23} = 0, a_{11}a_{21} + a_{12}a_{22} + a_{13}a_{23} = 0, a_{11}a_{21} + a_{12}a_{22} + a_{13}a_{23} = 0 \tag{A.13}$$

[7] http://plain.isas.jaxa.jp/ ebisawa/TEACHING/2007Komaba/2007Komaba/node7.html

が成立する。また、4.11、4.12、4.13 と e_1、e_2、e_3 の内積を取ることにより以下がわかる。

$$e_1' \cdot e_1 = a_{11}, e_1' \cdot e_2 = a_{12}, e_1' \cdot e_3 = a_{13} \tag{A.14}$$

$$e_2' \cdot e_1 = a_{21}, e_2' \cdot e_2 = a_{22}, e_2' \cdot e_3 = a_{23} \tag{A.15}$$

$$e_3' \cdot e_1 = a_{11}, e_3' \cdot e_2 = a_{32}, e_3' \cdot e_3 = a_{33} \tag{A.16}$$

となる。つまり、式 4.14 で定義される変換行列の 9 つの変換係数は旧座標系の 3 軸と新座標系の 3 軸の間のなす 9 つの角度の余弦の対応している。
よって、4.11、4.12、4.13 の逆変換は、

$$e_1 = a_{11}e_1' + a_{12}e_2' + a_{13}e_3' \tag{A.17}$$

$$e_2 = a_{21}e_1' + a_{22}e_2' + a_{23}e_3' \tag{A.18}$$

$$e_3 = a_{31}e_1' + a_{32}e_2' + a_{33}e_3' \tag{A.19}$$

となる。行列表示すると、

$$\begin{pmatrix} e_1 & e_2 & e_3 \end{pmatrix} = \begin{pmatrix} e_1' & e_2' & e_3' \end{pmatrix} \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \tag{A.20}$$

式 2.25 と比較すると、変換行列の行と列を入れかえた転置行列が逆行列になっていることがわかる。

A.1.3 オイラー角

座標変換を表す 3 つのパラメータとして良く使われるものにオイラー角[8] がある。オイラー角にも色々な定義があるが、日本の科学衛星の姿勢に使われる「xyz」オイラー角の定義を用いる。
赤道座標上で z 軸の周りに、+z の向きを向いて時計回りに角度 ϕ 回転し、x、y 軸の位置を変える。この新たな 3 軸を X, Y, Z(z=Z である) とする。次に Y 軸の周りに角度 θ 回転し、X、Z 軸の位置を変え、X', Y'(Y=Y'), Z' 軸を定義する。最後に Z' 軸の周りに ψ 回転し、最終的に X", Y", Z" 軸を定義することができる。
旧座標系と、X", Y", Z" 軸による新座標系の間の関係を与える（ϕ, θ, ψ）をオイラー角と呼ぶ。
定義したオイラー角と、変換行列との関係を示すと、まず最初の z 軸周りの ϕ 回転で新たな基底ベクトルと元の基底ベクトルの関係は以下のようになる。

$$\begin{pmatrix} e_1' & e_2' & e_3' \end{pmatrix} = \begin{pmatrix} e_1 & e_2 & e_3 \end{pmatrix} \begin{pmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{A.21}$$

同様に、Y 軸周りの θ 回転によって、

$$\begin{pmatrix} e_1'' & e_2'' & e_3'' \end{pmatrix} = \begin{pmatrix} e_1' & e_2' & e_3' \end{pmatrix} \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} \tag{A.22}$$

[8] http://plain.isas.jaxa.jp/ ebisawa/TEACHING/2007Komaba/2007Komaba/node13.html#EulerMatrix

Z' 軸周りの ψ 回転によって、

$$\begin{pmatrix} e_1''' & e_2''' & e_3''' \end{pmatrix} = \begin{pmatrix} e_1'' & e_2'' & e_3'' \end{pmatrix} \begin{pmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{A.23}$$

以上から 3 式をまとめると、

$$\begin{pmatrix} e_1''' & e_2''' & e_3''' \end{pmatrix} = \begin{pmatrix} e_1 & e_2 & e_3 \end{pmatrix} \begin{pmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{A.24}$$

となる。

A.2 「あかり」データのヘッダ部

```
  IMPLE = T / This is a FITS file
BITPIX = -32 /
NAXIS = 2 /
NAXIS1 = 6800 / NUMBER OF ELEMENTS ALONG THIS AXIS
NAXIS2 = 3600 / NUMBER OF ELEMENTS ALONG THIS AXIS
EXTEND = T / This file may contain FITS extensions
EQUINOX = 2000.0000 / Mean equinox
RADECSYS= 'FK5 ' / Astrometric system
CTYPE1 = 'GLON-MOL' / WCS projection type for this axis
CUNIT1 = 'deg ' / Axis unit
CRVAL1 = 0.000000000E+00 / World coordinate on this axis
CRPIX1 = 3.400500000E+03 / Reference pixel on this axis
CD1_1 = -5.000000000E-02 / Linear projection matrix
CD1_2 = 0.000000000E+00 / Linear projection matrix
CTYPE2 = 'GLAT-MOL' / WCS projection type for this axis
CUNIT2 = 'deg ' / Axis unit
CRVAL2 = 0.000000000E+00 / World coordinate on this axis
CRPIX2 = 1.800500000E+03 / Reference pixel on this axis
CD2_1 = 0.000000000E+00 / Linear projection matrix
CD2_2 = 5.000000000E-02 / Linear projection matrix
EXPTIME = 0.000000000E+00 / Maximum equivalent exposure time (s)
GAIN = 0.000000000E+00 / Maximum equivalent gain (e-/ADU)
COMMENT
SOFTNAME= 'SWarp ' / The software that processed those data
SOFTVERS= '2.16.0 ' / Version of the software
SOFTDATE= '2006-08-09' / Release date of the software
SOFTAUTH= 'Emmanuel BERTIN < bertin@iap.fr >' / Maintainer of the software
SOFTINST= 'TERAPIX team at IAP http://terapix.iap.fr' / Institute
COMMENT
```

AUTHOR = 'ircscan ' / Who ran the software
ORIGIN = 'cava5.ir.isas.jaxa.jp' / Where it was done
DATE = '2007-06-28T01:50:31' / When it was started (GMT)
COMBINET= 'MEDIAN ' / $COMBINE\_TYPE$ config parameter for SWarp
COMMENT
COMMENT Propagated FITS keywords
COMMENT
COMMENT Axis-dependent config parameters
RESAMPT1= 'BILINEAR' / RESAMPLING_TYPE config parameter
CENTERT1= 'MANUAL ' / CENTER_TYPE config parameter
PSCALET1= 'MANUAL ' / PIXELSCALE_TYPE config parameter
RESAMPT2= 'BILINEAR' / RESAMPLING_TYPE config parameter
CENTERT2= 'MANUAL ' / CENTER_TYPE config parameter
PSCALET2= 'MANUAL ' / PIXELSCALE_TYPE config parameter
END

今回使用した「あかり」入力 FITS ファイルのヘッダ部を示した。CTYPE1,CTYPE2 に座標が書かれており、デジタルプラネタリウム投影用の変換の際はこの部分は WCS は動作しない。

A.3 使用したプログラム

今回使用たプログラムを以下に示す。ここで使われている補間法はニアレストネイバー法である。

```
#include < stdio.h >
#include < stdlib.h >
#include < string.h >
#include </usr/local/xray/cfitsio/fitsio.h >
#include </usr/local/xray/cfitsio/longnam.h >
#include < wcshdr.h >
#include < wcs.h >
#include < math.h >

  main(int argc, char **argv){
FILE *fp;
char infile[6][256], gapfillfile[256];
char inputlist[256], PRJ[4];
char outfile[256], RAproj[9], DECproj[9];
long outimagesizeX, outimagesizeY;
fitsfile *fptr_in, *fptr_in2,*fptr_out;
int status=0, ncards, relax, nreject, nwcs, ctrl, i, j, anynul; long naxes[2];
float *inimage, *inimage2,*outimage, X_in, Y_in, X_out, Y_out; char *header;
struct wcsprm *wcs_in,*wcs_out;
double pixcrd_in[1][2], imgcrd[1][2], phi[1], theta[1], world[1][2]; double crpix1, crpix2, pixcrd_out[1][2];
int i_out, j_out, nfile, i_in, j_in;
```

```
double pixelsize;
long xsize, ysize;

   if(argc !=9){
printf("usage: convertWCS PRJ outimagesizeX outimagesizeY crpix1 crpix2 pixelsize infitsname
outfitsname n");
printf("where PRJ must be one of the following; CAR, AIT, ZEA, CSC, TSC, PCO. n");
exit(1);
}else
{
strcpy(PRJ, argv[1]);
sscanf(argv[2], "%ld", &outimagesizeX);
sscanf(argv[3], "%ld", &outimagesizeY);
sscanf(argv[4], "%lf", &crpix1);
sscanf(argv[5], "%lf", &crpix2);
sscanf(argv[6], "%lf", &pixelsize);
strcpy(infile[0], argv[7]);
strcpy(outfile, argv[8]);
}
printf("input file list = %s n", inputlist);
if(strcmp(PRJ,"CAR")!=0 &&strcmp(PRJ,"AIT")!=0&&strcmp(PRJ,"ZEA")!=0&&strcmp(PRJ,
"CSC")!
=0&&strcmp(PRJ, "PCO")!=0&&strcmp(PRJ, "TSC")!=0){
printf("PRJ must be one of the following: CAR, AIT, ZEA, CSC, PCO, TSC n");
exit(1);
}

   /* Read the input image ascii list */
/* if(NULL==(fp=fopen(inputlist,"r"))){
printf("cannot open %s n", inputlist);
exit(1);
}
*/
outimage=malloc(outimagesizeX*outimagesizeY*4);
for(i=0;i < outimagesizeX;i++){
for(j=0;j < outimagesizeY;j++){
outimage[i*outimagesizeY+j]=0.0;
}
}
/* put the output wcs keywords in the structure */
relax=1; /* 0 means recognize only FITS keywords defined by the
published WCS standard. 1 is more benign.*/
ctrl =0; /* If 0, not report any rejected header cards.
ctrl=1,2,3 gives detailed error messages in this order */
```

```
/* Create the output FITS file */
if (fits_create_file(&fptr_out, outfile, &status)) {
fits_report_error(stderr, status);
return 1;
}
naxes[0]=outimagesizeX; naxes[1]=outimagesizeY;
fits_create_img(fptr_out, FLOAT_IMG, 2L, naxes, &status);
fits_write_key_str(fptr_out, "RADESYS", "FK5", "Coordinate system", &status);
fits_write_key_dbl(fptr_out, "EQUINOX", 2000.0, 3, "Equinox",&status);
sprintf(RAproj, "RA—%s", PRJ);
fits_write_key_str(fptr_out, "CTYPE1", RAproj, "DEC projection",&status);
sprintf(DECproj, "DEC–%s", PRJ);
fits_write_key_str(fptr_out, "CTYPE2", DECproj, "DEC projection",&status);
fits_write_key_dbl(fptr_out, "CRPIX1", crpix1, 13, "X reference pixel",&status);
fits_write_key_dbl(fptr_out, "CRPIX2", crpix2, 13, "Y reference pixel",&status);
fits_write_key_dbl(fptr_out, "CRVAL1", 0.0, 13, "RA of reference pixel",&status);
fits_write_key_dbl(fptr_out, "CRVAL2", 0.0, 13, "DEC of reference pixel",&status);
fits_write_key_dbl(fptr_out, "CDELT1", -pixelsize, 13, "X pixel increment (degree)",&status);
fits_write_key_dbl(fptr_out, "CDELT2", pixelsize, 13, "Y pixel increment (degree)",&status);

   /* read the output FITS header */
if (fits_hdr2str(fptr_out, 1, NULL, 0, &header, &ncards, &status))
fits_report_error(stderr, status);
}

   /* put the output wcs keywords into the "wcs_out" structure */
if (status = wcspih(header, ncards, relax, ctrl, &nreject, &nwcs, &wcs_out)) {
fprintf(stderr, "wcspih ERROR %d: %s. n", status, wcs_errmsg[status]);
}
if (status = wcsset(wcs_out)) {
fprintf(stderr, "wcsset ERROR %d: %s. n", status,
wcs_errmsg[status]);
}
wcsprt(wcs_out);
free(header);
/* while(fscanf(fp, "%s", infile)==1)*/
/* for(nfile=0; nfile <=5; nfile++){*/
for(nfile=0; nfile <=0; nfile++){
/* open the input FITS file */
if (fits_open_file(&fptr_in, infile[nfile], READONLY, &status)) {
fits_report_error(stderr, status);
return 1;
}
printf("####### file %d: %s n", nfile+1, infile[nfile]);
```

```
/* read the input FITS header */
if (fits_hdr2str(fptr_in, 1, NULL, 0, &header, &ncards, &status))
fits_report_error(stderr, status);
return 1;
}
/* Put the input image into the image array */
if(nfile¡=1){
xsize=6800; ysize=3600;
inimage=malloc(xsize*ysize*4);
fits_read_2d_flt(fptr_in, 0, 0.0, xsize, xsize, ysize, inimage, &anynul, &status);
}else{
inimage=malloc(2600*3400*4);
xsize=2600; ysize=3400;
fits_read_2d_flt(fptr_in, 0, 0.0, 2600, 2600, 3400, inimage, &anynul, &status);
}
/* put the input wcs keywords in the structure */
if (status = wcspih(header, ncards, relax, ctrl, &nreject, &nwcs, &wcs_in))
fprintf(stderr, "wcspih ERROR %d: %s. n", status, wcs_errmsg[status]);

free(header);

  if (status = wcsset(wcs_in))
fprintf(stderr, "wcsset ERROR %d: %s. n", status,
wcs_errmsg[status]);

wcsprt(wcs_in);

  /* In the case of pieces_p5b/S4.fits and S6.fits, which have "gaps",
we need fill these gaps using pieces_p4b/S4.fits and S6.fits.*/
if(nfile¡=1)
inimage2=malloc(xsize*ysize*4);
if(nfile==0) strcpy(gapfillfile, "/Users/ogura/work/planetarium/AKARI5/ebisawa_prog/AllSky_S_p4.fits");
fits_open_file(&fptr_in2, gapfillfile, READONLY, &status);
fits_read_2d_flt(fptr_in2, 0, 0.0, xsize, xsize, ysize, inimage2, &anynul, &status);
if(nfile==0)
for(i=2043;i <=2379;i++)/* This is the gap region in the original p5b file*/
for(j=1855;j <=2149;j++)
inimage[i+j*xsize]=inimage2[i+j*xsize]; /* replace the part of p5b with p4b */
/* printf("debug %f  n", inimage[i+j*xsize]);*/ }
}
}
if(nfile==0)
for(i=5233;i <=5513;i++){/* This is the gap region in the original p5b file*/
for(j=1478;j <=1772;j++){
```

```
inimage[i+j*xsize]=inimage2[i+j*xsize];/* replace the part of p5b with p4b */
}
}
}
free(inimage2);
fits_close_file(fptr_in2, &status);
}
/* For loop for the output image pixels */
for(i=0;i < outimagesizeX;i++){
for(j=0;j < outimagesizeY;j++){
/* Define the image coordinages (X,Y), where (1,1)
is the lower left and X increases to left, Y increases
to up, then X=j+1, Y=i+1. */
X_out = (float)(i)+1.0; Y_out=(float)(j)+1.0;
/* Calculate the world coordinate for each output image pixel
using wcslib. */
pixcrd_out[0][0]=(double) X_out; pixcrd_out[0][1]=(double) Y_out;
wcsp2s(wcs_out,1,2,pixcrd_out[0],imgcrd[0],phi,theta,world[0],&status);
if(status==0){
/* printf("debug1 %f %f %f %f n",pixcrd_out[0][0],pixcrd_out[0][1],world[0][0],world[0][1]);*/ if(strcmp(wcs_in-
¿lngtyp,"GLON")==0 && strcmp(wcs_in-> lattyp,"GLAT")==0){
double l,b;
/* if the input image is written in the Galactic coordinates, we need to convert
(RA,DEC) of the output image to (l,b) of the input image */
radec2gal(world[0][0], world[0][1], &l, &b);
world[0][0]=l;
world[0][1]=b;
/* printf("debug2 %f %f n", l,b);*/
}
/* Calculate the pixel coordinates of the corresponding input image, either Galactic
or RA and DEC*/
wcss2p(wcs_in,1,2,world[0],phi,theta,imgcrd[0],pixcrd_in[0],&status);
/* somehow the X-pixel coordinate can become negative, measured from the
right end. In this case, we add the X-image size
if(pixcrd_in[0][0]¡0.0)pixcrd_in[0][0]=pixcrd_in[0][0]+xsize-1;}
*/
X_in=(float) pixcrd_in[0][0]; Y_in=(float) pixcrd_in[0][1];
if(X_in >=0.5&&X_in < xsize+0.5&&Y_in >=0.5&&Y_in < ysize+0.5) {
/* X_in and Y_in are the pixel coordinates where the lower-left corner of the
image is defined (0.5, 0.5) and the upper-right corner of the image
is (ximagesize+0.5, yimagesize+0.5).
Now we are going to average the four pixel values which surround X_in and Y_in.
+———+———+
— — —
```

```
— — —
— * — * — X in the left figure has the coordinate (X_in, Y_in).
— (i1,j2) — (i2,j2) —
— — X —
+———+———+
— — —
— — —
— * — * —
— (i1,j1) — (i2,j1) —
— — —
+———+———+
*/ int i1, i2, j1, j2;
int x1, y1;
int x2, y2;
i1 = max(0,(int)floor(X_in)-1);j1 = max(0,(int)floor(Y_in)-1);
i2 = min(xsize-1,i1+1); j2=min(ysize-1,j1+1);

   x1 =(int)(X_in+0.5);
y1 =(int)(Y_in+0.5);

   x2 = equ(i1,i2,x1);
y2 = equ(j1,j2,y1);

   outimage[i+j*outimagesizeX]=inimage[x2+y2*xsize];
% /* if(Y_in >=2052.551&&Y_in <=4091.449 && outimage[i+j*outimagesizeX]==0.0) {
printf("debug3 %f %f %f %f %d %d %f n",X_out, Y_out, X_in, Y_in, x2, y2, outimage[i+j*outimagesizeX]);
} */
/* printf("debug4 %f  n", inimage[x2+y2*xsize]);
printf("debug5 %f %f %d %d %f n", world[0][0],world[0][1],X_in, Y_in,outimage[i
+j*outimagesizeX]);*/
}
}
}
}
/* close the input file */
fits_close_file(fptr_in, &status);
printf("closing the input file status = %d n", status);
wcsfree(wcs_in);
}
fits_write_2d_flt(fptr_out, 0L, outimagesizeX, outimagesizeX, outimagesizeY, outimage, &sta-
tus);
/* close the output file*/
fits_close_file(fptr_out, &status);
wcsfree(wcs_out);
```

```
}
int radec2gal(double ra, double dec, double *l, double *b){
/* RA,DEC to Galactic conversion.
See http://plain.isas.jaxa.jp/ ebisawa/TEACHING/2007Komaba/2007Komaba/node16.html */
double x, y, z, X,Y,Z;
double deg2rad=1.745329252e-2;
x=cos(ra*deg2rad)*cos(dec*deg2rad);
y=sin(ra*deg2rad)*cos(dec*deg2rad);
z=sin(dec*deg2rad);
X = -0.0548755*x-0.8734370*y-0.483835*z;
Y = 0.4941100*x-0.4448300*y+0.746982*z;
Z = -0.8676660*x-0.1980760*y+0.455984*z;
*l=atan2(Y,X)/deg2rad;
if(*l¡0)*l=*l+360.0;
*b=atan(Z/sqrt(X*X+Y*Y))/deg2rad;
return;
}
int gal2radec(double l, double b, double *ra, double *dec){
/* Galactic to RA,DEC conversion.
See http://plain.isas.jaxa.jp/ ebisawa/TEACHING/2007Komaba/2007Komaba/node16.html */
double x, y, z, X,Y,Z;
double deg2rad=1.745329252e-2;
x=cos(l*deg2rad)*cos(b*deg2rad);
y=sin(l*deg2rad)*cos(b*deg2rad);
z=sin(b*deg2rad);
X = -0.0548755*x+0.494110*y-0.867666*z;
Y = -0.8734370*x-0.444830*y-0.198076*z;
Z = -0.4838350*x+0.746982*y+0.455984*z; *ra=atan2(Y,X)/deg2rad;
if(*ra¡0)*ra=*ra+360.0;}
*dec=atan(Z/sqrt(X*X+Y*Y))/deg2rad;
return;
}
int max(int a, int b){
if (a > b) {return a;}
else{return b;}
}
int min(int a, int b){
if (a < b) {return a;}
else{return b;}
}
int equ(int a, int b, int c){
if (a == b){return a;}
else{return b;}
}
```

A.3.1 プログラム変更点（バイリニア補間、バイキュービック補間）

使用したプログラムの補間法について変更した部分を示す。

バイリニア補間

```
   int i1, i2, j1, j2;
double xleft, xright, ydown, yup;
i1 = max(0,(int)floor(X_in)-1);j1 = max(0,(int)floor(Y_in)-1);
i2 = min(xsize-1,i1+1); j2=min(ysize-1,j1+1);
xleft = X_in - floor(X_in);
ydown = Y_in - floor(Y_in);
xright= floor(X_in+1.0) - X_in;
yup = floor(Y_in+1.0) - Y_in;
outimage[i+j*outimagesizeX]=
xleft*ydown *inimage[i2+j2*xsize]+
xleft*yup *inimage[i2+j1*xsize]+
xright*ydown*inimage[i1+j2*xsize]+
xright*yup *inimage[i1+j1*xsize];
```

バイキュービック補間

```
   int i1, i2, i3, i4, j1, j2, j3, j4;
double x1, x2, x3, x4, y1, y2, y3, y4;
i1 = max(0,(int)floor(X_in)-1);
i2 = min(xsize-1,i1+1);
i3 = min(xsize-1,i2+1);
i4 = min(xsize-1,i3+1);
j1 = max(0,(int)floor(Y_in)-1);
j2 = min(ysize-1,j1+1);
j3 = min(ysize-1,j2+1);
j4 = min(ysize-1,j3+1);

x1 = 1.0 + X_in - floor(X_in);
x2 = X_in - floor(X_in);
x3 = floor(X_in) + 1.0 - X_in;
x4 = floor(X_in) + 2.0 - X_in;
y1 = 1.0 + Y_in - floor(Y_in);
y2 = Y_in - floor(Y_in);
y3 = floor(Y_in) + 1.0 - Y_in;
y4 = floor(Y_in) + 2.0 - Y_in;
```

```
outimage[i+j*outimagesizeX]=
x1*y1 *inimage[i1+j1*xsize] + x2*y1 *inimage[i2+j1*xsize]+
x3*y1 *inimage[i3+j1*xsize] + x4*y1 *inimage[i4+j1*xsize]+
x1*y2 *inimage[i1+j2*xsize] + x2*y2 *inimage[i2+j2*xsize]+
x3*y2 *inimage[i3+j2*xsize] + x4*y2 *inimage[i4+j2*xsize]+
x1*y3 *inimage[i1+j3*xsize] + x2*y3 *inimage[i2+j3*xsize]+
x3*y3 *inimage[i3+j3*xsize] + x4*y3 *inimage[i4+j3*xsize]+
x1*y4 *inimage[i1+j4*xsize] + x2*y4 *inimage[i2+j4*xsize]+
x3*y4 *inimage[i3+j4*xsize] + x4*y4 *inimage[i4+j4*xsize];
```

A.4 All Sky Viewerについて

All Sky Viewer[9]は、高幣俊之氏（ORIHALCON）が開発した多機能なドーム映像プレビューツールである。３D空間中のバーチャルなドームスクリーンに、今回作成した「あかり」全天画像や映像などの投影することが可能であり、ドーム内で見回した場合や俯瞰で見た場合など、全天周映像を多角的に確認することができる。また、ドーム内視点の他に、ドームマスター形式やキューブマップ形式の１面などでも表示することができる。
また、製品となっているPro版というものもあり、映像をムービーや静止画連番に書き出すことで映像形式のコンバーターとしても利用できる。さらに、マルチ投影やドーム投影環境などでサーバ/クライアント方式で複数同期実行させることで、入力映像をリアルタイムにスライスして確認できるビューアとしても活用できる。

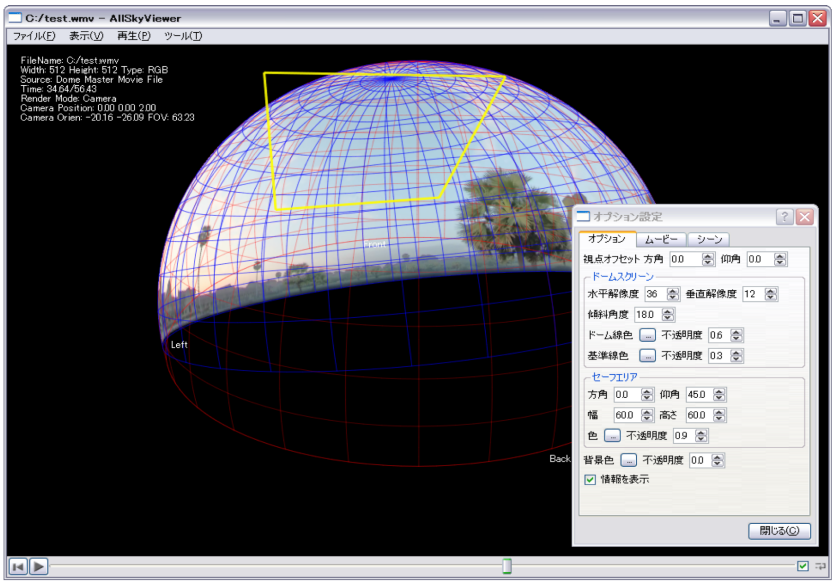

図 A.1: All Sky Viewer イメージ図

図 A.1 は All Sky Viewer のイメージ図である。以下に主な機能をまとめた。

[9]http://orihalcon.jp/projects/virtual-reality-environment/allskyviewer.html

- キューブマップ、ドームマスター、パノラマ形式の切り替え表示
- 様々な画像・動画フォーマットの読み込みに対応
- 仮想ドームスクリーンに投影した様子をリアルタイム表示
- スクリーンメッシュやセーフエリアなどの指標表示
- 実ドーム環境でのリアルタイムプレビュー（Pro 版）
- 任意サイズの画像・静止画連番・ムービーへの変換書き出し（Pro 版）
- コマンドラインによるバッチ変換・スライス処理の実行（Pro 版）

である。

A.5　ROSAT 全天サーベイデータへの応用

本研究で行った「あかり」データ用変換プログラムは、元の ROSAT データ変換プログラムが改良されている。4 章での記述のように ROSAT 全天サーベイデータは 1 枚の解像度が 512×512pixel のものが 1378 枚に分割されているため解像度が高い。すると、順変換を採用しても WCS の画素の算出が正しいように進み、画素の抜けがあまり目立たずに投影用フォーマットへ変換されているパターンであった。しかし実際は「あかり」データのように解像度が低い衛星観測データも存在することがあるので、補間法の導入できる逆変換が解像度の質を考慮しなくても投影用フォーマットへ変換でき、他の衛星観測データにも応用できる。そこで ROSAT データを「あかり」データ変換プログラムを用いて全天画像の作成を試みた。

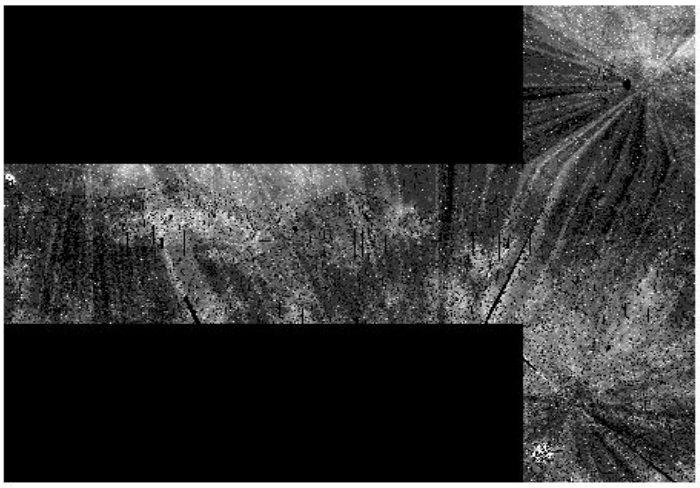

図 A.2: 改善プログラムを使用した ROSAT 全天画像

図 A.2 は「あかり」プログラムで走らせて作成した ROSAT データの全天画像である。ここで採用している補間法はニアレストネイバーで、わずかだが順変換プログラムよりも画像が改善できる。また、バイリニア補間、バイキュービック補間も試みているが、作成した画像は割愛する。

補間法	計算時間
ニアレストネイバー	約 1 日
バイリニア補間	約 2 日
バイキュービック補間	3 〜 4 日

表 A.1: ROSAT データ変換時の各補間法の計算時間

表 A.1 は各補間法の計算時間の比較である。ROSAT データは「あかり」データの 15 倍ほど解像度が高い（§4.3.2 参照）ため、計算時間もかなり長いことがわかった。他の観測データの応用から計算時間の短縮もこれからの課題となる。

関連図書

[1] 神原 弘之他, 2004, 電子情報通信学会誌, 87, 701-706
[2] 海老沢 研　宇宙科学講義ノート　2008 東京大学
[3] FITS の手引き　〜第 5.1 版〜　（天文情報処理研究会）
[4] E.W.Greisen&M.R.Calabretta.2002,Astronomy&Astrophysics, 395,1061-1075
[5] 久野 治義「赤外線工学」　電子情報通信学会
[6] Murakami et al. 2007, PASJ 59, S369-376
[7] Hidenori Takahashi., et al. 2000, SPIE, 4013, 47-58
[8] Hidenori Watarai., et al, 2000, SPIE, 4013, 59-68
[9] 西尾 光史　卒業論文（芝浦工業大学）2007
[10] E.BERTIN ,Institut d'Astrophysique & Observatoire de Paris, 2005
[11] 江尻 正員他　ディジタル画像処理　CG-ARTS 協会

表 目 次

図 目 次